TUGboat

Volume 17, Number 3 / September 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 330.

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.

Second-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/`

[D]esign is as integral to the print medium as words. Words are the substance; design is the form. Neither can communicate effectively without the other, and it is the synergy of well-written words and well-conceived design that makes print work.

Sean Morrison
A Guide to Type Design (1986)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 17, NUMBER 3 • SEPTEMBER 1996
SAN FRANCISCO • CALIFORNIA • U.S.A.

TUGboat

During 1996, the communications of the TeX Users Group will be published in four issues. The Proceedings of the 1996 TUG Annual Meeting appeared as Vol. 17, No. 2. The last issue (Vol. 17, No. 4) will be a theme issue, on TeX and the Humanities, with guest editors Christina Thiele and Pierre MacKay; participation is by invitation.

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 326.

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. 239).

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. 239).

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.

Trademarks

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

MS/DOS is a trademark of MicroSoft Corporation
METAFONT is a trademark of Addison-Wesley Inc.
PC TeX is a registered trademark of Personal TeX, Inc.
PostScript is a trademark of Adobe Systems, Inc.
TeX and AMS-TeX are trademarks of the American Mathematical Society.
Textures is a trademark of Blue Sky Research.
UNIX is a registered trademark of X/Open Co. Ltd.

1997 TEX Users Group Election

Barbara Beeton
for the Elections Committee

The terms of the TUG President and of 6 members of the Board of Directors expire as of the 1997 Annual Board Meeting, which will take place in conjunction with the 18th Annual TUG meeting to be held in July 1997 in San Francisco, California. The directors whose terms expire in 1997 are Robin Fairbairns, George Greenwade, Yannis Haralambous, Alan Hoenig, Jon Radel, and Sebastian Rahtz. The election to choose the new President and Board members will be held in Spring of 1997. Nominations for these openings are now being invited.

The Bylaws provide that "Any member may be nominated for election to the office of TUG President/to the Board by submitting a nomination petition in accordance with the TUG Election Procedures. Election ... shall be by written mail ballot of the entire membership, carried out in accordance with those same Procedures."

A change in the Election Procedures now provides for an election to be held only every second year, rather than every year. The term of the President will remain at two (2) years; the term of a director will change to four (4) years from three (3) effective with this election. Incumbent officers may be nominated for successive terms.

In order to align existing terms to the new schedule, the names of current directors whose terms expire in 1998 will, with their consent, be placed on the ballot for confirmation for one additional year.

The name of any member may be placed in nomination for election one of these offices by submission of a petition, signed by two other current (1997) members, to the TUG office at least two weeks (14 days) prior to the mailing of ballots. (A candidate's membership dues for 1997 will be expected to be paid by the nomination deadline.) A nomination form follows this announcement; forms may also be obtained from the TUG office, and electronically from the `usergrps/tug` area of CTAN or via the TUG Web pages `tug@tug.org`.

Along with a nomination form, each candidate is asked to supply a passport-size photograph, a short biography, and a statement of intent to be included with the ballot; the biography and statement of intent together may not exceed 400 words.

The deadline for receipt at the TUG office of nomination forms and ballot information is **March 4, 1997**.

Ballots will be mailed to all members in **March**. Marked ballots must be postmarked no later than **May 13**, and received no later than **May 27**. These deadlines will be noted on the ballots, and are included in the TUG Calendar in *TUGboat*.

Ballots will be counted by a disinterested party not part of the TUG organization. The results of the election should be available in early June, and will be announced in a future issue of *TUGboat* as well as through various TEX-related electronic lists.

1997 TUG Election — Nomination Form

Only current (1997) TUG members are eligible to participate. The signatures of two (2) members are required in addition to that of the nominee. **Type or print** names clearly, exactly as they appear on a TUG mailing label; new members should enter the name which they used on their membership application form. Names that do not exactly match the TUG records will not be accepted as valid.

The undersigned TUG members propose the nomination of:

Name of Nominee: ______________________

Signature: ______________________

Date: ______________________

for the position of (check one):

☐ **TUG President**

☐ **Member of the TUG Board of Directors**

for a term beginning with the 1997 Annual Meeting, **July 1997**.

Members supporting this nomination:

1. ______________________
(please print)

______________________ ____________
(signature) (date)

2. ______________________
(please print)

______________________ ____________
(signature) (date)

Return this nomination form to the TUG office (FAXed nomination forms will be accepted). Nomination forms and all required supplementary material (photograph, biography and personal statement for inclusion on the ballot) must be received in the TUG office no later than **March 4, 1997**.[1] It is the responsibility of the candidate to ensure that this deadline is met. Under no circumstances will incomplete applications be accepted.

☐ nomination form
☐ photograph
☐ biography/personal statement

TEX Users Group **FAX:** 415-982-8559
Nominations for 1997 Election
1850 Union Street, #1637
San Francisco, CA 94123
U.S.A.

[1] Supplementary material may be sent separately from the form, and supporting signatures need not all appear on one form.

General Delivery

From the President

The TUG Business Meeting at the 17th Annual TEX Users Group Meeting

As a TUG member you will have received the previous issue of *TUGboat*, which contains most of the papers presented at TUG'96, which took place in Dubna, 120 km north of Moscow, from July 28th to August 2nd 1996. I think this conference was a great success thanks to the hard work of all those involved, both organizers and participants. Let me take this occasion to express TUG's gratitude to Irina Makhovaya (*CyrTUG*'s Executive Director) and Prof. Evgueniy Pankratiev (President of *CyrTUG*) and their *CyrTUG* collaborators, and Dr. Vladimir Korenkov, vice-Director of the Laboratory of Computing Techniques and Automation of the Joint Institute of Nuclear Research (JINR, Dubna) and his team. Detailed travel reports (one by me and one by Robin Fairbairns, UKTUG's President) are available on the WWW (`http://tug.cs.umb.edu/tug96/newtug96.html` and `http://www.tex.ac.uk/UKTUG/TUG96.html`), and here I want to concentrate on matters discussed at the TUG Business Meeting on Monday July 29th 1996. In fact, six Board members were present in Dubna, and arrived two days before the conference started, so that we had the possibility to prepare a business plan that would guarantee that TUG could function optimally in the near future taking into account its limited financial resources.

TUG Business Meeting

I started the Business Meeting at 4 p.m. on Monday afternoon by introducing the members of the new Executive Committee: Yannis Haralambous (vice-President), Mimi Burbank (Treasurer), and Sebastian Rahtz (who is carrying on as Secretary).

I thanked the outgoing Board Members, Peter Flynn, former TUG Secretary, Michael Ferguson, former President of TUG's Technical Council, for their many years of dedicated work, and Mimi Jett, Tomas Rokicki and Norman Walsh for their contributions to TUG and the TEX Community. I expressed the hope that they would be able, in the future, to continue to contribute to the life of the TEX community.

Finally I announced the remaining Board members: Barbara Beeton, Karl Berry, Robin Fairbairns (Chair of UKTUG, co-opted for one year), George Greenwade, Judy Johnson, Jon Radel, and Jiří Zlatuška.

Since Michael Ferguson is no longer a Board Member, Sebastian Rahtz was appointed Chair of the Technical Council. To fill the vacancy in the Technical Council, it was suggested to have Alan Hoenig, well-known for his many interesting articles in *TUGboat*, serve on it; therefore I contacted Alan shortly after the Meeting in August and I am glad to announce that Alan accepted to serve on the Board and the Technical Council, initially for one year.

At the last TUG Business Meeting in St. Petersburg in Florida in 1995, I stated that I considered that TUG's main task was to get *TUGboat* back on schedule. Thanks to the hard work of the *TUGboat* production team (Barbara Beeton, *TUGboat* editor, Mimi Burbank, Team Coordinator, and Robin Fairbairns, Sebastian Rahtz, Christina Thiele and myself), with help from Malcolm Clark and Wietse Dol for some issues, we were able to produce four issues of our journal before Christmas 1995, and caught up with the normal production schedule before Easter of his year.

To meet the production schedule we, of course, had to make some compromises and for reasons of cost we have to limit ourselves to 96 pages per issue (paper cost has recently gone up by over 10%, domestic postage by about the same amount, while overseas postage rates have increased by about 30%!). We have folded TTN (TEX and *TUG News*) back into *TUGboat*, so that the information in *TUGboat* will contain a fair balance between news items, introductory, tutorial-level and more technically advanced articles.

Since last summer we have seen the success of the TDS (TEX Directory Structure) standard, which was adopted by most web2c distributions. The *TEX Live* CD-ROM has been published (as a collaboration between Thomas Esser, author of teTeX, GUTenberg, TUG, and UKTUG). It offers a plug-and-play TEX system for UNIX-based operating systems, and is built using the TDS layout. We hope to issue regular updated versions of this CD-ROM, possibly once a year. TUG and UKTUG also published the (long-promised) Edmac Manual.

To benefit maximally from the Internet, TUG now has its own server donated by Karl Berry and connected at UMB (University of Massachusetts at Boston). From that node we run a set of mailing lists (thanks to Peter Flynn who, until recently and for many years, looked after those lists on his computer in Cork, Ireland), a (limited) ftp service, and a WWW server. We hope to develop especially the WWW service `www.tug.org` in the near future in order to provide a unique entry-point for all TEX and TUG-related information. As many (especially N. American) TEX users will have experienced, SHSU has been withdrawn from service as a CTAN node. DANTE has donated the SparcStation that used to run the DANTE CTAN server in Heidelberg. The DANTE server is now running on a much larger and more powerful machine. Once the promised machine has arrived at Karl Berry's site (by early 1997, we hope), Karl and the CTAN maintainers will try to set up as fast as possible this long overdue replacement of a fully-supported CTAN node in North America. Our sincerest thanks go to DANTE for this generous offer.

Recently, illegally-changed Computer Modern font files were found to be present in some TEX distributions and on some public server sites. TUG strongly deplores the maintenance of out-of-date and corrupt data on publicly available resources and urges the maintainers to destroy the files in question as fast as possible. The Technical Council will look into the possibility of providing checking procedures to validate distributions in the future.

In order to have all TEX users in the world benefit from the articles in *TUGboat*, the Board decided to make all *TUGboat*s articles older then one year freely available electronically as fast as copyright clearance can be obtained from the various authors.

It is with great pleasure that I could acknowledge donations to the TUG'96 Bursary Fund by GUTenberg, NTG, UKTUG, TUG itself and a contribution in kind from DANTE. Also continued joint membership agreements with NTG and UKTUG are without doubt appreciated by people in the United Kingdom and the Netherlands, since it is more economical and simpler than separate memberships.

On the less positive side I had to mention that, notwithstanding a re-subscription campaign with members of the three previous years via email, membership numbers now stand at about 1600 (about half from North America), down again from last year's figures by about 15%. The increased cost of producing *TUGboat*, and the fixed cost of the TUG office probably will lead to a deficit of about \$20,000 for 1996. This means that drastic actions have to be taken to make sure that TUG can survive in the medium term, and various options have been considered for implementation early next year. Amongst these the reduction of the office staff to a half-time equivalent, and more reliance on email and WWW services will be introduced at the beginning of 1997, while other measures, still under discussion, are due to take effect by the summer of 1997. I hope that TUG will be able to count on your continued support to make this transition as smooth as possible, and that the financial situation will grow better and allow us to buy some equipment to develop the Internet services which we plan to offer soon.

At the end of the meeting some suggestions were made as to how TUG could guarantee the quality of TEX software, but lack of resources does not make this a viable possibility at present. Also, a more active role of the Technical Council in the development and the recognition of “TUG standards” was called for, and I am sure that the members of the TC will do their best to contribute in this area.

After the Meeting the Board decided that next year's TUG Conference will take place in San Francisco (see page 328) at the end of July, and I hope that many of you will be able to come and join us at TUG'97, near the cradle of TEX next year.[1]

Recently, the first public releases of Ω and e-TEX made it onto CTAN. As I explained already on several occasions, both these projects are extremely important for the survival of TEX in the medium and long term, since they show that TEX is not a frozen end-of-the-journey product. These developments, together with the new *tex2pdf* and *dvi2pdf* initiatives, described at TUG'96, show that the TEX community is very much alive and is well-prepared to address the challenges of text processing and document handling of the year 2000 (and beyond)!

As you will read on page 240, there are several openings for serving on the TUG Board for four years, starting in July 1997. We are also looking for candidates for the office of TUG President for a period of two years, also starting in July. Let me end by hoping that many of you care enough about the future of TUG and think that as an organization, hand in hand with the local TEX User sister-organizations, it still has an active role to play in the coming years, so that you will run for one of the open position mentioned, or contact us to volunteer for one of the many TUG-related tasks. It is only with your help and active support that we shall be able to guarantee that Knuth's work will continue to live until well after the turn of the century.

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

[1] We even have already two offers for hosting a TEX Conference in 1998! So, if you think that you could offer a site – and you have a lot of enthusiastic collaborators to help you make it all happen – please feel free to contact us at tug@tug.org.

Editorial Comments

Barbara Beeton

TeX Live: a CD-ROM for Unix

Through the efforts of Sebastian Rahtz and many others, sponsored by TUG, the UK TeX Users Group and GUTenberg (the French TeX users), with help from NTG (the Dutch TeX users), the CD *TeX Live* was launched in Paris at the end of May.

This CD contains a ready-to-run TeX setup — Thomas Esser's *teTeX*, based on Karl Berry's *Web2c*. There are binaries for ten different Unix platforms, plus archived packages of the GUTenberg Mac, DOS and Windows distributions, as well as the sources and a comprehensive (and carefully catalogued) set of macro packages and documentation.

The CD is arranged according to the TeX Directory Structure layout, as described in the report from the TUG TDS Working Group (*TUGboat* **16** (4), pp. 401–413).

More details, and ordering information, can be found at `http://www.tug.org/tex-live.html`. If you dont have WWW access, get in touch with the TUG office for details (e-mail to `tug@tug.org`, other contact information inside the front cover of this issue). The price is very modest, and all profits from sales go back to fund new versions of the CD, and to TeX-related development projects.

teTeX has been getting excellent reviews from users writing to `comp.text.tex`, and the *TeX Live* packaging provides a most convenient format.

CTAN update

After various trials and tribulations, the CTAN maintenance crew has removed the contents of the TeX archive from `ftp.shsu.edu` and removed that address from the list of maintained CTAN sites. (A current list can be had by requesting

```
   finger ctan@ftp.dante.de
or finger ctan@ftp.tex.ac.uk)
```

At present, the recommended site in the U.S. is `ftp.cdrom.com` on the West Coast. This site is a full mirror of the Cambridge site in the U.K. The root directory of the TeX portion of this archive is `/pub/tex/ctan`. This site does not currently support the "quote site index" option to search for particular file names, but for frequent users of the archive, the file `FILES.byname` in the TeX root area can be retrieved and searched for specific locations. This file is refreshed every night, so is a reliable resource.

Negotiations are underway to establish another primary CTAN site in the U.S.; developments will be reported in upcoming issues.

The new `ltugboat.cls`

We are pleased to announce the availability of a new LaTeX 2$_\varepsilon$ documentclass for *TUGboat* as well as a guide to its use; the latter appears in this issue under the authorship of Robin Fairbairns.

The documentclass has been constructed in the proper manner as a `tugboat.dtx` file with an associated `.ins` file to install it. These two files can be obtained from CTAN in the area `macros/latex/contrib/supported/tugboat/`, and are to be considered "official". The old `ltugboat.sty` file that occurs in various subdirectories at CTAN (apparently this has been a popular format for promulgating package documentation) should not be used to prepare items for submission to *TUGboat*.

Beta versions of this documentclass have been used to produce the last several issues of *TUGboat*, including the proceedings of TUG 96. We believe that it is well behaved when used with most existing LaTeX 2$_\varepsilon$ packages, but have obviously not tested all possibilities first-hand.

Of course, if the item you intend to submit to *TUGboat* is breaking new ground outside of LaTeX, the plain-based `tugboat.sty` (and `tugboat.cmn`) is still available: `macros/plain/contrib/tugboat/`; the user's guide is in `digests/tugboat/tubguide.tex`.

Copies of any of these files will be sent on request if you don't have internet access.

Frequently asked questions about TeX

It has long been a custom of network newsgroups to compile a FAQ, or list of "frequently asked/answered questions", to be consulted by newcomers to the list in order to avoid covering the same ground over and over again. The TeX community is no different in this regard, although it's perhaps a bit harder to find a current FAQ for TeX than for other groups. (The old version in the place where these files are usually located is seriously out of date, and not currently maintained.)

The best FAQ in English is the one put together by a working party of the UK TeX Users Group. It can be obtained as a file ready to print from CTAN in `usergrps/uktug/faq`. `newfaq.ps` is intended for A4 paper, and `letterfaq.ps` for U.S. lettersize paper. Alternatively, it can be searched interactively with a Web browser, at `http://www.cogs.sussex.ac.uk/cgi-bin/texfaq2html`.

A FAQ with a German slant is maintained by DANTE; it too is available from CTAN, in the area `usergrps/dante/de-tex-faq/`, or from the Web, `http://www.dante.de/dante/dante-faq.html`.

A French FAQ concentrates on LaTeX. It is posted monthly to `fr.comp.text.tex` and can be accessed via the Web from `http://www.loria.fr/tex/divers.html` (LaTeX Navigator) or by ftp from `ftp.inria.fr` in `/faq/comp.text.tex/FAQ_LaTeX_francaise`.

Pointers to other useful FAQ collections about TeX are welcome — I'll see that links are installed from the Web pages at both TUG and AMS.

Books about TeX, books prepared in TeX, and other bibliographies

It's time for another reminder about the extremely useful collection of bibliographies in BibTeX form compiled by Nelson Beebe. These comprise lists of publications *about* TeX, publications prepared *using* TeX, the complete contents of *TUGboat* (recompiled from the *TUGboat* tables of contents after every issue), and many, many other computer-related topics. They can be found at `ftp.math.utah.edu` in the directory `pub/tex/bibtex` or via the Web at `http://www.math.utah.edu/~beebe/#bibliographies`.

TeX output from a Web browser — Techexplorer

From IBM comes a plug-in hypermedia browser for viewing (La)TeX documents via Netscape Navigator on Windows 95 or Windows NT — Techexplorer. The current version is available at no charge from the IBM web site, `http://www.ics.raleigh.ibm.com/ics/techexp.htm`.

Robert Sutor, of IBM's Watson Labs, is the team leader for the product. He answers questions and sends regular reports to the list `techexplorer@listserv.nodak.edu`; they're well worth reading. To subscribe, send a message to `listserv@listserv.nodak.edu` with one line in the body:
`SUBSCRIBE TECHEXPLORER` your first and last names

Techexplorer does not use `dvi` files or METAFONT fonts; it dynamically renders TeX markup when it arrives, and uses the fonts available on the user's machine for the special symbols. If you're a PC user (or even if you're not — Unix support is being "actively investigated"), check it out!

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

Software & Tools

The Joy of TEX2PDF — Acrobatics with an Alternative to DVI Format

Petr Sojka, Han The Thanh, and Jiří Zlatuška

Abstract

This paper presents a discussion about generating Portable Document Format (PDF) directly from TEX source using a prototype TEX2PDF program. This is a derivative made from the TEX source which allows us to bypass DVI output generation, and to produce documents in Adobe PDF directly. Motivations for the TEX2PDF approach are discussed and further possible enhancements are outlined.

1 Motivation

GO FORTH *now*
and create masterpieces
of the publishing art!
Don Knuth [19], p. 303.

General acceptance of TEX for the publishing of technical documents has spread enormously during the last two decades. Since TEX's inception, however, new standards have emerged in the publishing world. SGML and LATEX for markup, PostScript and Portable Document Format as page description languages (PDL), are just a few of the buzzwords in the arena. Publishers are moving towards the art of creating *electronic* documents.

TEX's typesetting engine outputs its results in the device independent (DVI) page description format [9, 10]. To avoid duplication, and to be backward compatible, various extensions to the DVI format have been used via the `\special` command. Do you need color? Use color supporting `\specials`. Do you need PostScript fragments in the `.dvi` file? Graphics in various formats? PDF fragments in the `.dvi` file? Hypertext? Document/object structure markup for an SGML driver? Every new application usually ends up as a new set of `\specials`, which are unfortunately, not yet standardized [27, 25].

Do you need portable object reuse in your `.dvi` file? Sound? Portable Multiple Master font parameters? No `\specials` for these are in sight.

As a result of all this, documents in DVI format are not really portable, as they usually contain a lot of `\specials`, and visual appearance depends on the device drivers available at the reader's site. These

This paper was awarded the UK TEX User Group's Cathy Booth memorial prize at TUG '96 in Dubna, Russia.

and similar problems and thoughts have led us to research on the possibility of generating portable electronic documents which will offer widest range of functionality from well established and widely used (LA)TEX sources.

In section 2 we give an overview of current formats relevant to electronic document storage. In section 3 we discuss the current possibilities for producing PDF — a possible format of choice for electronic documents. We suggest a new approach by means of the TEX2PDF program in section 4 and, in section 5, its merits with respect to other approaches. We conclude with a discussion of object reuse in section 6, and future developments in section 7.

2 Formats for Electronic Document Delivery

2.1 DVI Format

A `.dvi` file is the standard output of a TEX run and is often used as a format for storage and exchange of typeset TEX documents.

DVI format is heavily (but not exclusively) used e.g. in the Los Alamos e-Print archive `http://xxx.lanl.gov/`. Several tens of thousands documents are available (typeset by autoTEXing scripts) from there. The disadvantage is that the documents are not 'self-embedded', which means that they rely on standardisation of font names and availability of fonts at the document consumer's site. Hypertext extensions to the DVI format have been accomplished by a set of HTML-like `\special`s defined by the HyperTEX project (`http://xxx.lanl.gov/hypertex/`) and special versions of previewers (`xhdvi`), `dvihps` and `ghostscript` (`ghosthview`) have been developed.

2.2 Portable Document Format

PDF [5] is a page description language derived from Adobe's PostScript language [2]. The design goals are:

- Rendering speed — algorithmic constructs were removed from the language.
- Portability — as a cross platform format, Acrobat Reader is available free of charge on major platforms.
- Compactness — the Lempel, Ziv, Welsh compression algorithm was licensed from UNISYS for maximum compression of files.[1] Multiple Master font technology, partial font downloading and built-in fonts in the Acrobat Reader lead to a minimum size for portable documents.
- WWW support — hypertext links to other documents on the Internet are allowed. PDF version 1.2 and Acrobat 3.0 (Amber) introduced a linearized arrangement of objects within PDF documents, allowing for incremental downloading across the Internet.
- Extensibility — documents can be extended without losing the old version; notes (stickers) can be added to a document by the readers.
- Password protection — access to a document can be protected by a password.
- Object structure — allows for access to individual pages, with possibility of one-pass generation.
- Easy exchange — ASCII (7bit) PDF files can be generated for better portability and email exchange.

PDF files can be embedded directly in an HTML page using the HTML `<EMBED>` tag [1]. These are becoming more and more popular in the WWW world, as they render faithfully what the author saw (modulo color rendering and resolution of an end-user's display).

2.3 SGML

Roll on SGML, and real document storage.
Not just this strange PDF thing
which traps the visuals like an insect in amber ...
James Robertson on `comp.text.pdf`

SGML is a widely accepted international standard (ISO 8879) [12, 6, 3] for document markup. It is the format of choice for document storage chosen by many publishers [23, 7, 4]. It is a language for describing markup, aimed at long-term storage, but not at visual layout. As TEX's typesetting engine is still the state-of-the-art, the perspective of typesetting of SGML documents via LATEX3 with a TEX based engine is a viable option.

3 Current Possibilities for Producing PDF from TEX

If PDF is required as the end format, with currently available programs one has to generate PostScript from a `.dvi` file and then to 'distill' (using Adobe's Distiller program) the result to PDF. Some comments and suggestions on how to create PDF files from TEX are collected in [17]. Problems with configuring fonts are described in [28] and [8].

[1] Latest news from Adobe says that ZIP compression has been added as well, leading to even better compression ratios.

4 The Name of the Game

There still are countless important issues to be studied, relating especially to the many classes of documents that go far beyond what I ever intended TEX to handle.
Don Knuth [21], p. 640

Motivated by a note by Don Knuth to one of the authors (private communication, 1994), who mentioned he expected people would attempt to create derivations from TEX suitable for, e.g., outputting PostScript instead of DVI, a project for creating PDF files directly from the TEX source has been attempted [14], introducing the possibility of creating either DVI or PDF output. The working name of this game is TEX2PDF. An example of the TEX source taking advantage of the new possibilities is shown in figure 1 and the resulting document as viewed with Adobe Acrobat Reader is shown in figure 2 on page 251.

4.1 New primitives

New primitives have been introduced in TEX2PDF in order to allow for more straightforward use of hypertext features from within TEX-like source. Most of their parameters are taken implicitly from the context of use in TEX terms, which simplifies their use considerably. We do not specify the full syntax here, because it is not yet fully stable.

`\pdfoutput` changes TEX2PDF behaviour from DVI-producing mode to PDF-producing one.

`\pdfannottext` takes an argument which specifies the text of an annotation to be created at the current position.

`\pdfannotlink`, `\pdfendlink` allows the user to specify hypertext links with all of the link attributes available in the PDF specification. An integer argument is used as a key to the corresponding anchor. If no link border has been specified, it is computed for all boxes between `\pdfannotlink` and `\pdfendlink`, so the link will automatically become multiline if a line break occurs in between.

`\pdfoutline` allows for the generation of bookmarks; bookmarks can be hierarchically structured.

`\pdfdestxyz`, `\pdfdestfit`, `\pdfdestfith`, `\pdfdestfitv` provide specification of various types of anchors with zooming and fitting possibilities.

`\pdfdestfitr`, `\pdfendfitr` specify the position of anchor corners. In this case, the anchor area is computed from the corners.

4.2 Font handling

Font handling in TEX2PDF is currently limited to Type1 fonts only. Metric information is extracted from the `pfb` file. Font name mapping is handled using an auxiliary font mapping configuration file introducing the list of fonts available, together with the information on the type of font embedding and its usage.

Virtual fonts [18] are supported in TEX2PDF. As they are in fact part of `.dvi` files, they have to be unfolded before PDF is output, as in today's DVI drivers.

4.3 Compression

Compression is allowed in the PDF specification, and several types of compression filters can be used; JPEG compression for color graphics, LZW and ZIP compression for text and graphics, and CCITT Group, Run Length and LZW compression for monochrome images.

As the LZW compression algorithm is licensed by UNISYS, we cannot distribute TEX2PDF with LZW support, but we used it for testing runs to compare TEX2PDF with Distiller (see table on the following page). However, the even more effective ZIP compression will be available in PDF version 1.2, avoiding the need for LZW compression in TEX2PDF, and the patent problems. The test figures show that TEX2PDF generated an even more compact PDF file than Adobe Distiller on standard text files.

4.4 Graphics

`\special`s are not yet handled by TEX2PDF. As most of the graphics included in TEX documents are PostScript and TIFF, at least support for the PostScript to PDF and TIFF to PDF conversion will have to be included in the future.

4.5 Implementation

The implementation of of TEX2PDF is realized as a `web` change file to the latest TEX source [20]. This implies that TEX2PDF is as portable as TEX itself is. Karl Berry's `web2c` package has been used for the development and for producing a running UNIX version. We expect easy recompilation on any UNIX platform.

5 Pros and Cons

I was constantly bombarded by ideas for extensions, and I was constantly turning a deaf ear to everything that did not fit well with TEX as I conceived it at the time.
Don Knuth [21], p. 640

To compare TeX2PDF with the other methods of producing a hypertext PDF document from a TeX file, we did several testing runs. They were done on a Sun Sparc 10 under the Solaris 2.4 operating system. Measurements were done using the `time` program (CPU times are listed). We used `tex.tex`, generated from the TeX source (`tex.web`) file, as the testing document. For the hypertext version we used a slightly changed version of `webmac.tex` (see `http://www.cstug.cz/~thanh/tex2pdf`).

In both time and size comparisons TeX2PDF beats its competitors (see tables on the following page). This is mainly due to the absence of intermediate DVI and PostScript formats in TeX2PDF, allowing for better PDF optimisation. TeX2PDF is slightly slowed down by `pfb` file parsing.

The users familiar with the (`emacs` + TeX + `xhdvi` (+ `ghostscript`)) suite of programs might want to switch to (`emacs` + TeX2PDF + `xpdf`), thus speeding up the document debugging cycle considerably.

TeX2PDF is written in `web` so that its source blends naturally with the source of TeX the program. The obvious benefit is absolute compatibility with TeX proper; the actual code which drives the typesetting engine is that of Don Knuth (modulo `whatsits` use for the hypertext primitives added in TeX2PDF). While this conformance to TeX source greatly benefits from Don's appreciation of stability, it makes the implementor's life more difficult in the world where PDF still evolves. It is also hard to debug TeX2PDF without incremental compilation. When we come to add implementation of `\special` commands, maintenance will become tough.

The changes introduced in new versions of PDF are motivated by achieving better performance when handling Acrobat documents, and so TeX2PDF is bound to have the PDF-generating modules modified or rewritten so that maximum benefit of the features supported by PDF technology can be used. The fact that PDF specification has been made public is crucial to success of this approach.

The TeX2PDF approach is naturally backward compatible with TeX — in fact, if PDF output is not switched on, it can still generate DVI output identical to that of TeX. Just by redefining some cross-referencing macros, the new hypertext features of TeX2PDF can be instantly used even without modifying the markup of old LaTeX documents.

6 Object Reuse

Using well-designed formats results in LaTeX source that clearly reflects the document structure.
T. V. Raman [24]

With PDF, there is the possibility of taking advantage of the object structure and manipulation specified within a PDF file to store elements of document structure (higher level document model) in the PDF file generated by the application (TeX2PDF). Some work has been already done in this direction by defining Encapsulated PDF (EPDF) blocks and their reuse [26]. This format, however, is not supported or used by a wide variety of applications.

The logical structure of a document model is also urgently needed in applications like AsTeR [24], which *reads* LaTeX documents using a speech synthesizer. Developing an application that is capable of reading aloud enriched PDF files might become possible.

Our suggestions for further work could lead to primitives which allow handling of PDF *objects* stored in the trailer of a PDF file indirectly. At least three primitives are foreseen:

`\setpdfbox` typesets its argument and stores the result as a PDF object. The reference to that object will stay in the internal register accessible by `\lastpdfbox`.

`\lastpdfbox` returns the reference to the last stored object by `\setpdfbox`.

`\usepdfbox` This primitive puts a *reference* to an object into the output stream.

7 Future Work

Few claim to know what will be the preferred electronic format a century from now, but I'm willing to go out on a limb and assert that it will be none of TeX, PostScript, PDF, Microsoft Word, nor any other format currently in existence.
Paul Ginsparg [11]

TeX2PDF is currently under development and is available to beta testers only. We do not guarantee that the input syntax will remain unchanged. Support for object reuse, graphics and OpenType (TrueType) fonts when the PDF specification 1.2 comes out may be added.

For testing purposes, a `tex2pdf` option for the `hyperref` package [16] will be written, using the hypertext possibilities of TeX2PDF directly. This will allow using TeX2PDF for re-typesetting of LaTeX documents just by loading with `hyperref`

Program(s)	Time without compression	Time with compression (LZW)
TEX2PDF (α-test version)	1:57	2:38
TEX + dvips 5.58 + Adobe Distiller 2.1	6:34 (1:33+0:18+4:43)	6:56 (1:33+0:18+5:05)
TEX + dvips 5.58 + Aladdin Ghostscript 4.0	40:23 (1:33+0:18+38:32)	not applicable

Table 1: Speed comparison of several ways of producing PDF file (tex.pdf) from a TEX file (tex.tex)

Program(s)	without LZW compression	with LZW compression	without compression and PDF file gzipped
TEX2PDF (α-test version)	8 063 658	3 086 545	1 906 184
TEX + dvips 5.58 + Adobe Distiller 2.1	10 530 967	4 387 232	2 115 827
TEX + dvips 5.58 + Aladdin Ghostscript 4.0	16 908 552	not applicable	

Table 2: Size comparison of several ways of producing PDF file (tex.pdf) from a TEX file (tex.tex)

package with the `tex2pdf` option in the document preamble.

Support for the full usage of Multiple Master technology remains to be added, possibly in the combination with METAFONT [15, 13]. Extensions of the paragraph breaking algorithm [22] to take advantage of Multiple Master fonts with a variable width axis (but constant grayness) to help justification (`\emergencyfontwidthstretch`) is another possible direction of future work.

Acknowledgements

The support of the TUG '96 bursary committee is acknowledged, having allowed presentation of a preliminary version of this paper at the TUG '96 conference in Dubna, Russia.

References

[1] Adobe. Adobe Acrobat 3.0 beta. http://www.adobe.com/acrobat/3beta/main.html, 1996.

[2] Adobe Systems. *PostScript Language Reference Manual*. Addison-Wesley, Reading, MA, USA, 1985.

[3] American National Standards Institute and International Organization for Standardization. *Information processing: Text and Office Systems: Standard Generalized Markup Language (SGML)*. American National Standards Institute, 1430 Broadway, New York, NY 10018, USA, 1985.

[4] Association of American Publishers. *Association of American Publishers Electronic Manuscript Series Standard for Electronic Manuscript Preparation and Markup: an SGML Application Conforming to International Standard ISO 8879–Standard Generalized Markup Language. Version 2.0 Dublin, Ohio: Available from the Electronic Publishing Special Interest Group, c1987.* Association of American Publishers, Dublin, OH, USA, 1987.

[5] Tim Bienz, Richard Cohn, and James R. Meehan. *Portable Document Format Reference Manual, Version 1.1*. Addison-Wesley, Reading, MA, USA, 1996.

[6] Steven J. DeRose and David G. Durand. *Making Hypermedia Work*. Kluwer Academic Publishers Group, Norwell, MA, USA, and Dordrecht, The Netherlands, 1994.

[7] Andrew E. Dobrowolski. Typesetting SGML Documents using TEX. *TUGboat*, 12(3):409–414, December 1991.

[8] Emerge, Inc. TEX and PDF: Solving Font Problems. http://www.emrg.com/texpdf.html, 1996.

[9] David Fuchs. The Format of TEX's DVI Files. *TUGboat*, 1(1):17, October 1980.

[10] David Fuchs. The Format of TEX's DVI Files. *TUGboat*, 3(2):14, October 1982.

[11] Paul Ginsparg. Winners and Losers in the Global Research Village. http://xxx.lanl.gov/blurb/pg96unesco.html, February 1996.

[12] Charles F. Goldfarb and Yuri Rubinsky. *The SGML Handbook*. Clarendon Press, Oxford, UK, 1990.

[13] Michel Goossens, Sebastian Rahtz, and Robin Fairbairns. Using Adobe Type 1 Multiple Master Fonts with TEX. *TUGboat*, 16(3):253–258, June 1995.

[14] Han The Thanh. Portable Document Format and Typesetting System TEX (in Czech). Master's thesis, Masaryk University, Brno, April 1996.

[15] Yannis Haralambous. Parametrization of Postscript Fonts through METAFONT—an Alternative to Adobe Multiple Master Fonts. *Electronic Publishing*, 6(3):145–157, April 1994.

[16] Yannis Haralambous and Sebastian Rahtz. LaTeX, Hypertext and PDF, or the Entry of TeX into the World of Hypertext. *TUGboat*, 16(2):162–173, June 1995.

[17] Berthold K. P. Horn. Acrobat PDF from TeX. `http://www.YandY.com/pdf_from.pdf`, 1996.

[18] Donald Knuth. Virtual Fonts: More Fun for Grand Wizards. *TUGboat*, 11(1):13–23, April 1990.

[19] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[20] Donald E. Knuth. *TeX: The Program*, volume B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[21] Donald E. Knuth. The Errors of TeX. *Software–Practice and Experience*, 19(7):607–685, 1989.

[22] Donald E. Knuth and Michael F. Plass. Breaking Paragraphs into Lines. *Software–Practice and Experience*, 11:1119–1184, 1981.

[23] Sebastian P. Q. Rahtz. Another Look at LaTeX to SGML Conversion. *TUGboat*, 16(3):162–173, September 1995.

[24] T. V. Raman. An Audio View of TeX Documents. *TUGboat*, 13(3):372–379, October 1992.

[25] Tomas G. Rokicki. A Proposed Standard for Specials. *TUGboat*, 16(4):395–401, December 1995.

[26] Philip N. Smith. Block-Base Formatting with Encapsulated PDF. Technical Report NOTTCS-TR-95-1, Department of Computer Science, University of Nottingham, January 1995. `http://www.ep.cs.nott.ac.uk/~pns/pdfcorner/complete.pdf`.

[27] Mike Sofka. Dvi Driver Implementation and Standardization Issues. Available as `http://www.rpi.edu/~sofkam/DVI/dvi.html`, 1996–.

[28] Kendall Whitehouse. Creating Quality Adobe PDF Files from TeX with Dvips. `http://www.adobe.com/supportservice/custsupport/SOLUTIONS/2d7a.htm`, 1996.

◇ Petr Sojka, Han The Thanh, and Jiří Zlatuška
Faculty of Informatics
Masaryk University Brno
Burešova 20, 602 00 Brno
Czech Republic
Internet: `sojka, thanh, zlatuska@informatics.muni.cz`

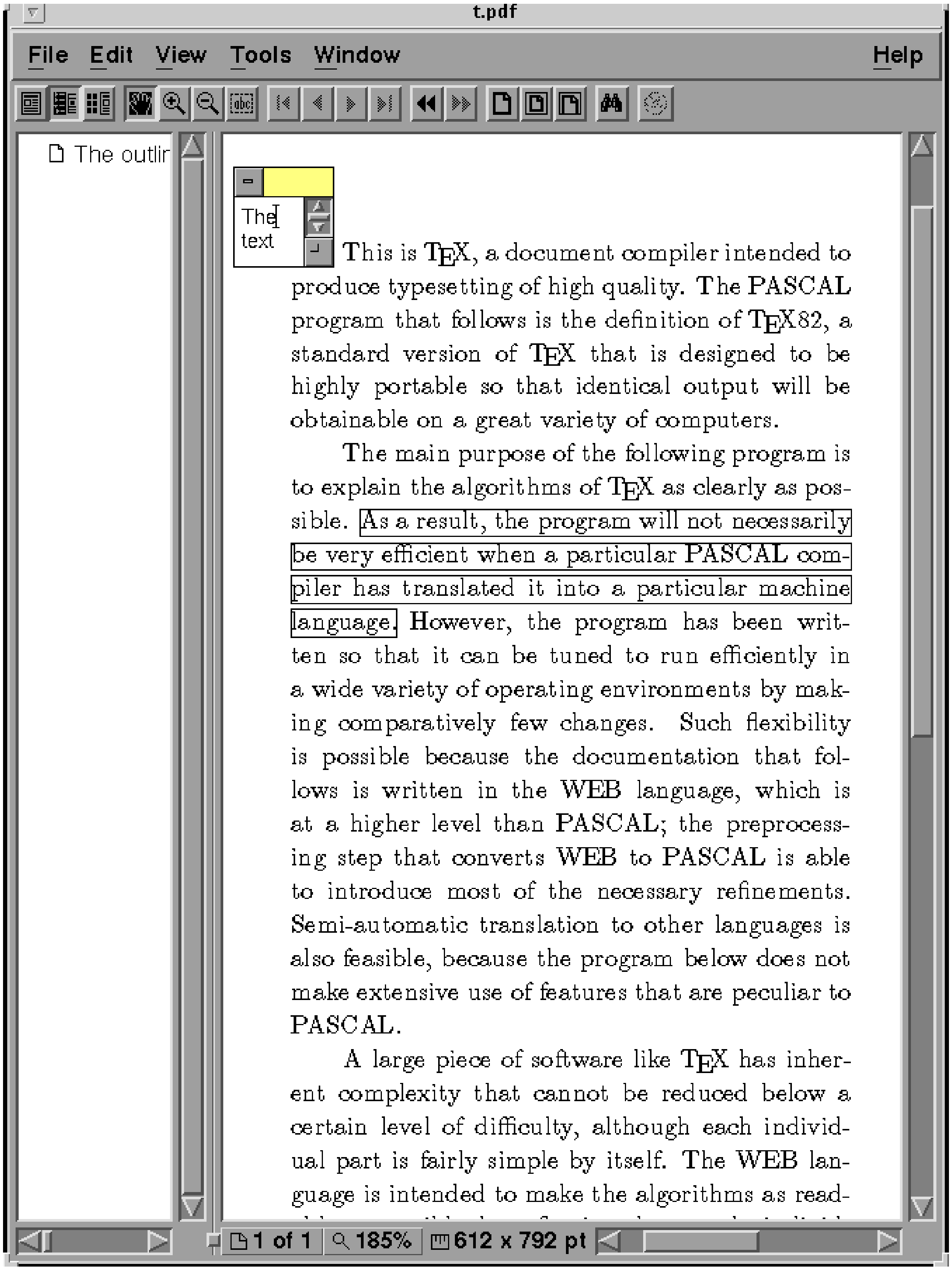
```
%% LaTeX2e file `t.tex'
%%
\hsize 3in
\baselineskip 13pt
\pdfoutput=1              % we will produce PDF instead of DVI
\pdfannottext
    open                  % optional specification if the text annotation is implicitly opened
    {The text annotation} % the text itself
\def\BL{\pdfannotlink
    depth 3pt height 8pt  % optional specification for link size
    1                     % key of destination
    border 0 0 1          % optional specification for link border
}
\def\EL{\pdfendlink}
\pdfoutline
    1                     % key of destination
    0                     % number of sub-entries of this item
    {The outline entry}   % Text of this item
\pdfdestxyz
    1                     % key of this destination
    zoom 2                % optional zoom factor
%\pdfdestfit  1 or %\pdfdestfith 1 or %\pdfdestfitv 1
%\pdfdestfitr 1 ... \pdfendfitr

This is \TeX, a document compiler intended to produce typesetting of
high quality.  The PASCAL program that follows is the definition of
\TeX82, a standard version of \TeX\ that is designed to be highly
portable so that identical output will be obtainable on a great
variety of computers.

The main purpose of the following program is to explain the algorithms
of \TeX\ as clearly as possible. \BL As a result, the program will not
necessarily be very efficient when a particular PASCAL compiler has
translated it into a particular machine language.\EL\ However, the
program has been written so that it can be tuned to run efficiently in
a wide variety of operating environments by making comparatively few
changes. Such flexibility is possible because the documentation that
follows is written in the WEB language, which is at a higher level
than PASCAL; the preprocessing step that converts WEB to PASCAL is
able to introduce most of the necessary refinements.  Semi-automatic
translation to other languages is also feasible, because the program
below does not make extensive use of features that are peculiar to
PASCAL.

A large piece of software like \TeX\ has inherent complexity that cannot
be reduced below a certain level of difficulty, although each individual
part is fairly simple by itself. The WEB language is intended to make
the algorithms as readable as possible, by reflecting the way the
individual program pieces fit together and by providing the
cross-references that connect different parts. Detailed comments about
what is going on, and about why things were done in certain ways, have
been liberally sprinkled throughout the program.  These comments explain
features of the implementation, but they rarely attempt to explain the
\TeX\ language itself, since the reader is supposed to be familiar with
{\sl The \TeX book}.
\bye
```

Figure 1: Example of new hypertext primitives added in the TEX2PDF source file



This is TEX, a document compiler intended to produce typesetting of high quality. The PASCAL program that follows is the definition of TEX82, a standard version of TEX that is designed to be highly portable so that identical output will be obtainable on a great variety of computers.

The main purpose of the following program is to explain the algorithms of TEX as clearly as possible. As a result, the program will not necessarily be very efficient when a particular PASCAL compiler has translated it into a particular machine language. However, the program has been written so that it can be tuned to run efficiently in a wide variety of operating environments by making comparatively few changes. Such flexibility is possible because the documentation that follows is written in the WEB language, which is at a higher level than PASCAL; the preprocessing step that converts WEB to PASCAL is able to introduce most of the necessary refinements. Semi-automatic translation to other languages is also feasible, because the program below does not make extensive use of features that are peculiar to PASCAL.

A large piece of software like TEX has inherent complexity that cannot be reduced below a certain level of difficulty, although each individual part is fairly simple by itself. The WEB language is intended to make the algorithms as read-

Figure 2: Result of TEX2PDF source in Fig. 1 viewed in Acrobat Reader

The DVIPDF Program

Sergey Lesenko

Abstract

This article describes the DVIPDF program, a DVI driver producing as its output Portable Document Format. It reviews the current state of development of the program and makes some suggestions for `\special` syntax.

1 Introduction

Among the common problems in the world of TeX is the question of how to produce documents with hypertext capability and high-quality printing at the same time; the Portable Document Format (PDF)[1] by Adobe permits us to resolve this particular issue. There are currently several ways to generate PDF output:

- TeX $\longrightarrow$ DVIPS [4] $\longrightarrow$ Adobe Distiller
- TeX $\longrightarrow$ DVIPS $\longrightarrow$ GhostScript [2]
- TeX2PDF [5]
- TeX $\longrightarrow$ DVIPDF

The most frequently used solution (the first case above) permits us to generate the most functionally complete PDF files, thanks to the *hyperref* package [3]. This route can be compared with the DVIPDF program. Since it is based on Distiller (the commercial product from Adobe), the former may stay ahead in terms of features; since the PDF specification has recently been upgraded, there is need for further development after shipment of the new version of Distiller.

But the DVIPDF way has its effective side too. If we consider the process of getting output from a `.dvi` file, then we have only the following step:

Step One: Translation from DVI to PDF

But the process using Distiller has two steps:

Step One: Translation from DVI to PDF

Step Two: Translation from PDF to PDF

In the latter case there is a loss of precision for characters, rules and other objects because intermediate values are used in the `.ps` file, not those actually present in the `.dvi` file. DVIPDF makes it possible to generate output with high precision, although currently with a limited set of features.

[1] Described in [1], and also available from `http://www.adobe.com/supportservice/devrelations/devtechnotes.html` and `ftp://ftp.adobe.com/pub/adobe/Applications/Acrobat/SDK/TECHDOC/PDFSPEC.PDF`.

2 Current features and those in development

DVIPDF is based on DVIPS by Tomas Rokicki, and in future may be integrated with it, I would like to hope. What can the DVIPDF program do now and what will it be able to do in future? The current version supports the following features:

- Rotated and scaled text;
- Rotated and scaled graphics (BMP and JPEG formats);
- Colors for text and background;
- Annotations and bookmarks;
- HTML links and links to other PDF files;
- Partial font loading;
- Reencoding.

At present only two graphic formats are supported. The BMP format allows the insertion of illustrations in the PDF file with black & white (1-bit), gray (8-bit) and color (24-bit) models. The JPEG format allows gray and color models. The capabilities for text and background colors correspond to those in DVIPS. Geometric transformation of text as a graphic object may be nested up to sixteen times in any way desired. As far as the hypertext capabilities are concerned, annotations may be nested; for bookmarks this is limited to six levels.

Embedded fonts are PostScript Type 1, using partial font downloading. Re-encoding can be performed on internal (embedded) fonts as well as external (referenced) fonts. Use of external fonts decreases the size of the output, but the potential user has to have these fonts available.

The most important problems for future development are:

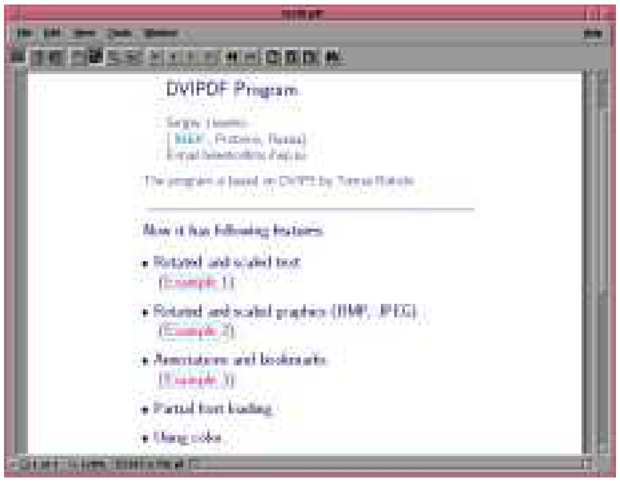
- Support for Encapsulated PostScript illustrations;
- Support for new features of the PDF 1.2 specification.

There is currently a way to insert EPS (by producing BMP using GhostScript, and then inserting the figure in BMP format), but the result is not scalable.

There are no plans to support all features of PDF; for example, bitmapped fonts will not be addressed, since they render very badly with the Acrobat Reader; on the other hand, features such as thumbnails may be added, but only much later."

3 Suggestions for `\special` syntax

Since PDF has some unique features, I had some problems choosing the optimal variants of `\special`



DVIPDF Program

Sergey Lesenko
(IHEP, Protvino, Russia)
E-mail lesenko@mx.ihep.su

The program is based on DVIPS by Tomas Rokicki

Now it has following features

- Rotated and scaled text
  (Example 1)
- Rotated and scaled graphics (BMP, JPEG)
  (Example 2)
- Annotations and bookmarks
  (Example 3)
- Partial font loading
- Using color

Figure 1:

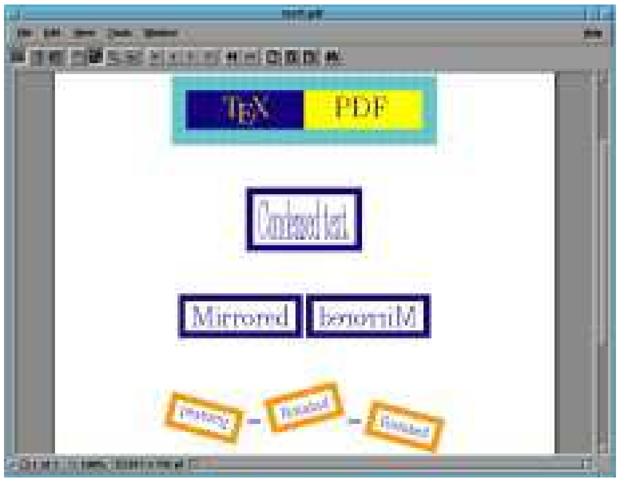


Figure 2:

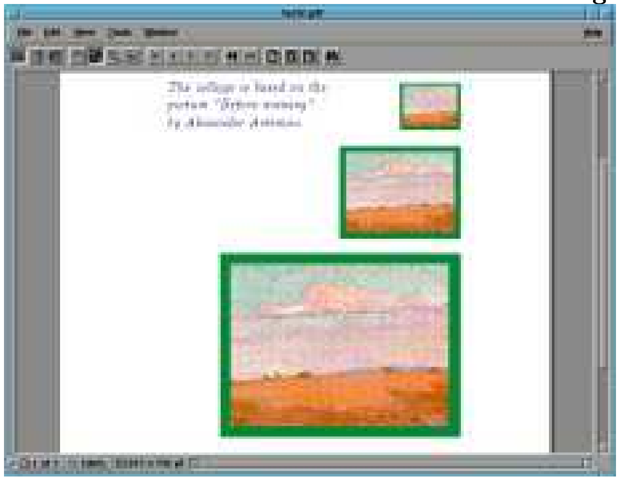

The collage is based on the picture "Before evening" by Alexander Antonov

Figure 3:

commands. Let us consider two variants and call them *universal* and *pdf*:

- *universal* to support existing `\special` conventions;
- *pdf* to be oriented only to PDF output.

The *universal* `\special` permits us to generate either a `.ps` file or a `.pdf` file from the same `.dvi` file. This is very useful, if we distribute only the `.dvi` file without TEX sources. However, the *universal* `\special` may need some information only for PostScript output, and some destined solely for Distiller. This leads to redundancy and increased size of the `.dvi` file.

The *pdf* `\special` is more compact and simpler for parsing; since PDF is developing as a standard format, it seems that this second way is preferable, and this was what I have implemented in the current version of DVIPDF.

I would like to introduce some suggestions for syntax:

- `pdf:` – first token to identify a *pdf* `\special`;
- `/ABC` – token consisting only of uppercase characters for definition of type;
- `/Abc` or `/abc` – token for subtype;
- `Abc` or `abc` – alphabetic parameters;
- `123` or `123.000` – numeric parameters;
- `<<` and `>>` – tokens to mark `push` and `pop`.

The above syntax description makes for simple parsing of the `.dvi` file. Some examples are offered in the following table:

Begin rotation	pdf: /ROT 30 <<
End rotation	pdf: /ROT >>
Begin scaling	pdf: /SC 4.0 2.0 <<
End Scaling	pdf: /SC >>
Begin color	pdf: /C Blue <<
End color	pdf: /C >>
Begin annotat.*	pdf: /ANN /LNK /Dest test <<
End annotation	pdf: /ANN >>
Graphics	pdf: /GRAPH filename 123 123

* here some secondary parameters are omitted

I would like to suggest that we pass size parameters for graphics in scaled points (*sp*), not big points (*bp*), since DVIPDF deals with *sp* when illustrations are inserted into some object and then scaled or rotated; recall that *bp*, as a unit, is used only for producing output. The program simply calculates new coordinates and does not need to worry about converting *bp* into *sp*.

To then estimate the effectiveness of the *pdf* `\special` set, I produced DVI output for some TEX files with different `\special` commands. These results are presented in the following table. To generate this data, I used the same sources as for producing the PDF slide files discussed in the next section.

File	*universal*	*pdf*
TEST0.DVI	3200	2276
TEST1.DVI	4244	2400
TEST2.DVI	2676	1720
TEST3.DVI	7284	3808

We can see that `.dvi` files with the *pdf* `\special` format are more compact.

4 Some results

To demonstrate the results I would like to present some figures from slides which were prepared for TUG'96.[2] The collection consists of four `.pdf` files (they are DVIPDF output): main file (`TEST0.PDF`) as a menu, and three auxiliary files (`TEST1.PDF`, `TEST2.PDF` and `TEST3.PDF`).

The main file has one HTML link ($\boxed{\text{IHEP}}$ on Fig. 1) and three links ($\boxed{\text{Example 1}}$, $\boxed{\text{Example 2}}$ $\boxed{\text{Example 3}}$ on Fig. 1) to auxiliary files.

When we click on the HTML link, the Acrobat Reader passes a request to our browser (Netscape, for example) and it asks for the IHEP home page (if, of course, our computer is connected to the Internet).

If we click on any link to auxiliary files, the chosen file will be loaded and the Reader will view it. Each file has a link ($\boxed{\text{Return}}$ on Fig. 2, Fig. 3) to the main file, so that we can return to the main file and examine the other auxiliary files.

If all goes well, I hope to release DVIPDF for testing towards the end of 1996.

Acknowledgements

I would like to thank Michel Goossens and Sebastian Rahtz for supporting this project, and Mimi Burbank and Sebastian Rahtz for editing this paper.

References

[1] Tim Bienz and Richard Cohn. *Portable Document Format Reference Manual.* Addison Wesley, Reading, MA, 1993.

[2] L. Peter Deutsch. Aladdin ghostscript, 1996. version 4.01, available at `ftp://ftp.cs.wisc.edu/pub/ghost/aladdin`.

[3] Yannis Haralambous and Sebastian Rahtz. LATEX, hypertext and PDF, *or* the entry of TEX into the world of hypertext. *TUGboat*, 16(2):162–173, June 1995.

[4] Tomas Rokicki. Dvips: A TEX driver, 1994. version 5.58, available at `ftp://labrea.stanford.edu/pub/tex`.

[5] Petr Sojka and Han The Thanh. The Joy of TEX2PDF — Acrobatics with an alternative to DVI format. *TUGboat*, 17(3), 1996.

◇ Sergey Lesenko
P.O. Box 35
Institute for High Energy Physics
Scientific Information Department
Protvino, Moscow Region, Russia
`lesenko@mx.ihep.su`

[2] 17th Annual Meeting of the TEX Users Group in Dubna, Russia, July 28th – August 2th, 1996

TEXtensions

Editing .dvi Files, or Visual TEX

Jonathan Fine

Abstract

This note outlines the specification of a TEX format, that will allow the resulting `.dvi` file to be edited via a suitable previewer and a `.dvi` file editor. Such close linking of editing and typesetting appears to be within the present capabilities of TEX.

Value Added Typesetting

Typesetting can be thought of as a process which adds value to the document being processed. This may not be true for works typeset from the author's original manuscript and corrected proofs, for such physical documents reveal change of mind, history of composition and other details which are lost in the printed version of the document. But here we consider the typesetting of, say, a suitably tagged ASCII file.

Throughout this document we will use the language and conventions of TEX, but most of the issues involved are of a more general nature, and apply to any computer typesetting system.

Suppose throughout that `myfile.tex` is typeset to produce `myfile.dvi`. If the latter file is the former, together with some added value, then it should be possible to recover the former from the latter. Oddly enough, a recent posting to an electronic discussion list raised precisely this problem. An author had in error deleted the original `.tex` file, and wished to recover its content, as best as was possible, from the `.dvi` file. This then is the definition of *value added typesetting*—from the typeset file it must be possible to extract the source file.

Poppelier (1991) also contains a discussion of the process by which typesetting adds value to the document, but from a different point of view.

Specials

TEX has a process by which special instructions can be transmitted to printing devices (*The TEXbook*, page 226), and that is the `\special` mechanism. Each `\special` that makes it to the `.dvi` file will produce in it a string of characters, attached to some specific location on the page. These characters are not usually intended to be printed on the page, rather their purpose is to control the printing process in some way.

This mechanism can be used to embed some text into the `.dvi` file, but one should (*The TEXbook*, page 228) "be careful not to make the list [of characters, i.e., the text] too long, or you might overflow TEX's string memory." The author does not know if this will be a danger for the constructions about to be described. Using emTEX he has created a `.dvi` file with 500,000 different specials, whose content is the numbers from 1 to 500,000, as digit strings.

One solution to the added value problem is to place the entire text of the input file `myfile.tex` as special in the document. Although this satisfies the formal requirement, it is a little coarse. To edit the file `myfile.dvi` consists of editing the copy of `myfile.tex` which is embedded as a special in `myfile.dvi`. It would not be difficult to adapt a text editing program, so that it operated on this embedded special, rather than a self-contained file. TEX can then be run, without an error arising one hopes, to refresh `myfile.dvi`.

Although coarse, this illustrates the essence of the method by which `.dvi` files may be edited via the previewer. What is required is that the process be refined.

So far as I know, products such as Lightning Textures continually refresh the previewed `.dvi` file as the the user changes the source `.tex` file, but do not associate the individual characters, words or markup in the underlying `.tex` file to the content of the displayed `.dvi` file. Thus, the user cannot edit the `.tex` file solely by interacting with the `.dvi` file. This is possible with Scientific Word, which should be thought of as a WSYIWYG or more exactly visual editor, whose underlying file format is LATEX. I believe that it is precisely because TEX as usually used does not allow the solution of the problems described here, that Scientific Word does not use `tex` to format files for the editor to display.

Smaller Specials

The text of a document, say as an ASCII file, is naturally broken down into paragraphs, words, characters, and spaces. It seems natural to break a document down into words. They are the smallest units of meaning. This is reflected in the very name of the tool used by authors to prepare documents, the word processor. Programmers are more accustomed to using the file editor.

For the moment, we shall assume that the document is very plain, with no changes of font or other control sequences. Suppose that we have a TEX format that will, besides typesetting the document, place before each word in the document a `\special`, whose content is the following word, as represented in the source file. Because of ligatures and hyphenation, this may not be the same as the characters which follow the `\special` in the `.dvi` file.

Suppose that `myfile.dvi` is created from `myfile.tex` by using this format. It will not be difficult, by extracting the text of the specials, to recreate `myfile.tex` from `myfile.dvi`. (This is not strictly correct. Assuming the usual category codes, additional spaces between words, additional lines between paragraphs, and the location of line breaks within paragraphs, will all be lost when passing from `myfile.tex` to `myfile.dvi`. This is probably no great loss. Contrarywise, typeset paragraph line breaks have been introduced. It may even be an advantage to have a source file whose line breaks agree with those of the `.dvi` file.)

A Special Format

It is not so difficult to create a format that will read the input file word by word, and place the words as it reads them into `\specials`. The code below, which is intended to be read in an environment where white space is ignored, and `~` is a space character, shows the basic features of such an environment.

The macro `\sentinel` is used simply to indicate the end of a paragraph, or the end of the file.

```
\def \sentinel { \noexpand \sentinel }
```

The idea now is to define `\dopar` so that

```
\dopar
The first paragraph is not very
long at all.

The second paragraph is even
shorter.

\sentinel
```

will result in appropiate typesetting and specials.

Here is a simple (too simple) implementation. The macro `\dopar` will read text paragraph by paragraph, until the `\sentinel` follows a blank line or explicit `\par`.

```
\def\dopar #1 \par #2
{
  \doword #1 ~\sentinel \par
  \if #2 \sentinel
    \let \next \relax
  \else
    \let \next \dopar
  \fi
  \next
}
```

The macro `\doword` similarly goes through the paragraph word by word until `\sentinel` is reached.

```
\def \doword #1~#2
{
  \special { #1 }  #1~
  \ifx #2 \sentinel
    \let \next \gobble
  \else
    \let \next \doword
  \fi
  \next #2
}
```

This sample code is not intended to be the basis for a practical implementation of a format that will create value-added `.dvi` files. Rather, its purpose is to show that such a format is possible, and to draw attention to some of the difficulties which may be encountered when creating such an object.

Editing via a Previewer

Suppose now that `myfile.dvi` has been created by a format file as above. The viewer notices a misspelt wrod. Within a special in the `.dvi` file, it is easy to change the letters `wrod` into `word`. It will be harder to add or delete letters within the special, because there would not be room for the addition at the correct point in the `.dvi` file, or a hole would be left in it. But this is the sort of problem which editing programs are accustomed to dealing with.

Now that the copy of `myfile.tex` which is within `myfile.dvi` has been changed, one would like the rest of `myfile.dvi` to be brought up to date. For simplicity, we shall assume that `myfile.dvi` is simply one page, a galley that is long enough to accommodate all that is placed on it. Changing `wrod` to `word` will change the paragraph in which it is placed. The change will in general be more complicated than replacing `wrod` by `word`. Even this simple change may change the line breaks in the paragraph. Hyphenation may change, as may ligatures and kerning. Correcting a simple letter transposition error will require resetting the whole paragraph.

In most TeX formats, the size and content of one paragraph does not influence the setting of the others. All then that needs to be reset is the paragraph in which the change occurred. If the document were set into pages rather than just a galley, then page breaks would also need to be reconsidered.

This discussion has focussed on changing the letters in a single word in a paragraph. Adding or deleting whole words will go the same way, as will addition or deletion of paragraphs, provided the format does not do something like numbering paragraphs. In any case, TeX will be required to process some text when a change is made to the `.tex` file embedded in the `.dvi` file, to bring the `.dvi` file up to date.

Calling TeX from the Previewer

When the user has finished making changes to the paragraph, or earlier if wished, the previewer must call upon TeX to reprocess the changed paragraph. As before, we assume that the difficulty faced by all editing programs, of deleting or adding material in the middle of a file, has been solved.

TeX turns a text file into a `.dvi` file. Ordinarily, this `.dvi` file is not accessible until TeX has come to an `\end`. By writing a suitable device or virtual file, which will depend on the operating system, it should be possible to use the `.dvi` output of TeX before the `\end`. Thus, a command such as

```
\shipout\vbox{\input tempfile}
```

will cause TeX to produce a page which contains the revised and reset paragraph, which the editing functions attached to the previewer can now paste into place, replacing the old version. Incidentally, if the output of TeX is a virtual `.dvi` file, then there should be no reason why the input `tempfile.tex` should not also be a virtual device.

The foregoing discussion is intended to demonstrate that by combining TeX as it is, together with a suitable format, a suitable previewer, editors for text and `.dvi` files, and a bit of operating system virtual file magic, it is possible to produce a WSYIWYG variant of TeX. Users of this composite program will be able to edit their documents via the previewer. The format as described is limited to a single page, a single font, and no mathematics, but it does have multiple paragraphs.

Breaking Pages

Suppose now that the document is broken into pages, and a paragraph is added. All subsequent pages, and perhaps the previous page, will have to be reconsidered. Assume TeX is being used in a normal manner, so that parameters such as `\linepenalty` and `\brokenpenalty` do not change from page to page. Were TeX to reprocess the whole of the changed `myfile.tex`, all paragraphs from the previous version would be broken exactly as before, and so there would be no need for them to be reset.

The previewer and editor combination can ask TeX to break the new document by passing it a suitably coded sequence of boxes, penalties, skips and kerns to break, for this is all the page breaking mechanism (*The TeXbook*, Chapter 15) operates on. Alternatively, some other program could be asked to do the breaking. This is the approach taken by Type & Set (Asher, 1992).

Control Words

The aspect which presents the most difficulties is now to be discussed, and briefly at that. Most documents contain control words, such as `\TeX` and `\section` and `\eqalign`. These are part of the source `myfile.tex` and so they must go into `myfile.dvi` as specials. When it comes to the editing process, even though the addition or deletion of a control word such as `\TeX` is fairly innocuous, to add or remove an `\equalign` can have a drastic effect.

The braces `{}`, and the mathematics shift character `$`, will also have a drastic effect when added or removed from `myfile.tex`. The same is true when the delimiter required by some macro is omitted. For this approach to have all the typesetting power and programmability which TeX provides, access to local change of font etc., parameter delimiters, and mathematics shift must be provided. This has to be done in a manner which is consistent with the editing requirements imposed by the embedded `\special` approach.

Unbalanced braces, missing or extra mathematics shift characters, and missing delimiters all provide difficulties for users of TeX. A format which detected such input errors early, before they gave rise to an error message from the stomach of TeX, could make TeX easier for many users. A format which allows the user to edit the copy of `myfile.tex` embedded within `myfile.dvi` would similarly have to detect input errors before they reach TeX's stomach.

Performance

It is quite possible that such a format, which reproduces the input text as specials, and detects all errors before TeX does, will run at perhaps a tenth of the speed of a regular format. However, the usual approach requires the document to be typeset as a whole, and so an unchanged paragraph may be typeset a dozen times or more during the revisions. A format which sets whole documents slower, but which is able to reset the document paragraph by paragraph may very well consume fewer machine cycles over the life of a document.

Moreover, the paragraphs can be set or reset as completed, rather than file by file. If the computer is sufficiently rapid, and many are today, this machine work can be done as required. This will result in the user being locked out for a short period at the end of each paragraph, while processing takes place, just as an editing program may deny access to the user while a file is being written.

Thinking alike

Since writing this article, I found the following statement put forward to motivate the CONCUR feature provided by SGML.

> It is sometimes useful to maintain information about a source and a result document simultaneously in the same document, as in "what you see is what you get" (WSYIWYG) word processors. There, the user appears to interact with the formatted output, but the editorial changes are actually made in the source, which is then reformatted for display.

This quotation comes from Annex C.3.1 of ISO 8879 (the SGML standard) and is also reproduced (as is the whole of ISO 8879) in Goldfarb (page 88).

Conclusions

More can be done with TeX as it is, than is commonly realised. Some of its limitations exist in the imagination of the critics rather than in the program itself. I hope that all those who say that TeX cannot do such-and-such think carefully as to why it is that TeX cannot do what they wish.

For a portable WSYIWYG variant of TeX to be produced, the various additional components, which are previewer, `.dvi` file editor, text file editor, and operating system magic, must also be portable or ported. An editor for `.dvi` files is probably the most important new program. For this to work most effectively, it may be necessary to extend the `.dvi` file specifications.

Also required is a format file, which satisfies requirements far more exacting than met at present. This also would be a major piece of work.

Acknowledgements

The author thanks Mimi Burbank for correcting many errors in a previous version of this article, and Alan Hoenig for some encouraging remarks.

Postscript

For various reasons the publication of this article has been long delayed. (It was widely circulated in preprint form at TUG'93 in Aston, England, and was submitted later in that year. Not all of the delay has been due to the author.) Since then, the author has solved all the fundamental problems involved in creating a TeX format file that will create a `.dvi` file that can be visually processed. Also since then the World Wide Web has 'taken off' and this provides an additional reason for producing `.dvi` files which support rich visual interaction. The article "TeX innovations at the Louis-Jean printing house", by Laugier and Haralambous, describes an interesting program that allows certain changes to be made interactively to a `.dvi` file. Also relevant is the author's own article, "Documents, compuscripts, programs and macros".

There is some overlap between this article and Rahtz, "Another look at LaTeX to SGML conversion". In both cases, the placing of text from the document into the `.dvi` file as `\special`s is a crucial technique. In this article the purpose is to store the original text of the article. For Rahtz, the purpose is to create a transformed form of the article.

Also relevant is the article of Kawaguti and Kitajima, which describes a different approach. They have created a loosely coupled composite from adaptions of two existing tools, namely `emacs` and `xdvi`. Special tags are added to a traditional (La)TeX source file, which produce `\special`s containing file name and line numbers in the `.dvi` file. These are used to support new `emacs` and `xdvi` commands, which together allow the user to move the `emacs` cursor by clicking on the previewed `.dvi` file. This is, as the authors recognise, not the same as what this article calls visual typesetting.

Would those who are interested in following the path outlined in the present article, particularly on the viewing and editing side, please contact me. For such a system to provide an open architecture, standards for the use of specials, going far beyond Rokicki's "A proposed standard for specials", will be required.

Bibliography

Asher, Graham. "Inside Type & Set", *TUGboat* **13** (1), pages 13–22, 1992.

Fine, Jonathan. "Documents, compuscripts, programs and macros", *TUGboat* **15** (3), pages 381–385, 1994.

Goldfarb, Charles F. *The* SGML *Handbook*, Oxford University Press, 1990

Kawaguti, Minato and Kitajima, Norio. "Concurrent use of an interactive TEX previewer with an Emacs-type editor", *TUGboat* **15** (3), pages 293–300, 1994.

Laugier, Maurice, and Yannis Haralambous, "TEX innovations at the Louis-Jean printing house", *TUGboat* **15** (4), pages 438–443, 1994.

Lavagnino, John. "Simultaneous electronic and paper publication", *TUGboat* **12** (3), pages 401–405, 1991.

Mittelbach, Frank. "E-TEX: Guidelines for future TEX", *TUGboat* **11** (3), pages 337–345, 1990.

Poppelier, N. A. F. M. "SGML and TEX in Scientific Publishing", *TUGboat* **12** (1), pages 105–109, 1991.

Rahtz, Sebastian. "Another look at LATEX to SGML conversion". *TUGboat* **16** (3), pages 315–324, 1995.

Rokicki, Tomas G. "A proposed standard for specials", *TUGboat* **16** (4), pages 395–401, 1995.

Taylor, Philip. "The future of TEX", *TUGboat* **13** (4), pages 433–442, 1992.

Siebenmann, Laurent. "The Lion and the Mouse", EuroTEX '92 Proceedings, edited by Jiří Zlatuška, pages 43–52, 1992.

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

Philology

TEX in Russia: ab ovo or About the TEXnical evolution in Russia

Irina A. Makhovaya

In the beginning was the word. And the word was "TEX".

The story goes that Donald Knuth studied Russian specifically for the purpose of reading the papers and monographs of Soviet mathematicians in their native language. When creating his famous program, the Grand Wizard must have been thinking about the poor Russian scientists who didn't have opportunities to publish their works in foreign journals. Perhaps this was one of the reasons for the name TEX: it's the first syllable of the Russian word TEKHNOLOGIYA ('technologia', or the Greek word, to be more precise). Knuth considered the proper pronunciation of the word so important that he alloted the first chapter of his book [27] to explain the right sound of his "child's" name. Fortunately, it is not a problem for Russian readers to do this.

The first non-Latin font in the Computer Modern family was the Cyrillic font `mcyr` [3, 4], created in 1985 as a 7-bit encoding font. The American Mathematical Society often translated the works of Soviet mathematicians into English in their journal, *Mathematical Reviews*, and included Soviet publications in its bibliography; `mcyr` made it possible to reproduce the actual titles and Soviet authors' names.

In 1989 D. Vulis [67] offered Russian TEX in an 8-bit encoding scheme; it combined T. Ridgeway's (University of Washington, Seattle) Cyrillic fonts and D. Vulis' hyphenation algorithm (based on the F. Liang algorithm).

So, it seemed preordained by the Grand Wizard and his successors that TEX should become part of Russian reality and a necessary software tool for our scientists.

And what about the situation in the Soviet Union at that time?

TEX made its appearance in the mid-1980s. At first, it was only known to the scientists who travelled to large scientific institutes in Western Europe and the USA. They were, as a rule, the physicists, including scientists from the Institute of High Energy

Physics (IHEP, in Protvino). The first Cyrillic version of TEX appeared at that very institute. It became known as "Protvinskaya" ('Protvinskaia'); the authors were S. Klimenko, B. Malyshev, A. Samarin, et al. [2, 5, 11, 12, 13, 14, 15, 21, 22, 23, 26, 43, 56, 57]. The first Cyrillic font in this version was `tt` [11, 12, 15]: according to the rules of thesis preparation at the time it was necessary to type them on a typewriter (not a computer!). Since the young scientists were lazy, they devised a way to deceive the bureaucrats and so did not retype materials that were prepared on computers.

By the end of the '80s and the beginning of the '90s, TEX had lost its exotic character. There were some specialists and the groups of scientists who began to use TEX with some Cyrillic fonts and to implement their own Cyrillic versions of TEX.

In an agreement between the American Mathematical Society and the Soviet publishing houses of Mir Publishers, Nauka Publishers and Leningrad State University, three Russian specialists were sent to the AMS for AMS-TEX training. During this visit, they called on the TUG office [55] and the idea to create *CyrTUG* in the USSR was born.

In the spring of 1991 (May 23–24), there was a "constituent assembly" of *CyrTUG* at Mir Publishers [46]. There were 23 TEX users from Moscow, Protvino, Saint-Petersburg and Novisibirsk at the meeting. The participants reported on their work on Cyrillic versions of TEXand the President, Executive Director, and Board were elected. The Cyrillic TEX Users Group (or, in Russian: Associaciĭa Polzovateleĭ Kirillicheskogo TEX'a) was born.

We all consider Dmitriı Vulis the main "culprit" behind this. He was in correspondence with most of us: V. Andrushchenko (Institute of Russian, Moscow), J. Romanovskiı (Saint Petersburg State University), A. Samarin (IHEP, Protvino), I. Makhovaya (Mir Publishers, Moscow), A. Urvantsev (Novosibirsk State University), et al. He acquainted us all with each other.

CyrTUG held its next annual meeting in Moscow at the Central Economics & Mathematics Institute (CEMI) October 20–22, 1992 [47]. The meeting was attended by 53 members of the group. There were (besides the above) the inhabitants of Irkutsk, Vladivostok, Syktyvkar, Sochi and Rostov-on-Don. Unfortunately, for financial reasons, we were not able to issue the proceedings of these conferences.

Since then, *CyrTUG* meetings have taken place every year; since 1993 (October 4–7 Pereslavl-Zalesskiı) our foreign colleagues have taken part in our conferences. J. Roseman (USA) and K. van der Laan (the Netherlands) were the first. K. van der Laan gave some talks and this promoted an extension of our users' horizon. He wrote about his impressions in a paper [31]. In 1994 (September 7–10, Dubna) M. Goossens (Switzerland) joined us [16, 32]; in 1995 (October 3–7, Protvino) there were four more — J. Roseman, M. Goossens, K. Pishka (Czech Republic) and B. Jackowski (Poland). We all can see the transformation of the process now in July 1996!

The pilgrimage originated in 1991 when T. Jurriens (the Netherlands) [24] gave a course to people from Novosibirsk State University TEX and D. Guenther (Washington State University) instructed TEX users from Moscow and Kazan at Mir Publishers.

The number of Russian TEX users who have gone abroad to work or take part in conferences has snowballed [1, 7, 8, 17, 18, 19, 25, 29, 30, 33, 34, 35, 39, 40, 41, 42, 44, 58, 61, 66]. TEX90 (Cork, Ireland, 1990) was the first meeting with a Russian representative, followed by two more Paris in 1991; Prague, 1992, six; Aston, 1993, two; Gdansk, 1994, two; Santa Barbara, 1995, one; and last year in Arnhem, 15 Russians! Not only the quantity but also the quality of Russian TEX users' work has increased. The main fields of interest remain Cyrillisation, fonts, encoding schemes and pictures.

For the five years *CyrTUG* has been in existence some 700 TEX users have been members, with about 50 scientific institutes, universities and publishing houses as institutional members. There have also been citizens of the USA, Slovak Republic, Czech Republic, Switzerland, and the Netherlands; institutional members include CERN (Geneva), JINR (Dubna), Mir Publishers (Moscow), MekhMath faculty of Moscow State University, UrbanSoft (St. Petersburg), Institute of Mathematics (Kazan), Electrotechnical Institute (Novosibirsk), Institute of Mathematics and Mechanics (Ekaterinburg), and others. The Grand Wizard, Donald Knuth, is an honorary member of *CyrTUG*: during his visit to St. Petersburg he was presented with card No. 0314 [48, 49].

We can see the increase of TEX popularity in Russia. The reason is not only active *CyrTUG* work but also the many monographs and textbooks in Russian which have been published [9, 27, 37, 38, 53, 63, 64], and a number of articles [20, 28, 65]. During this time there have also been a lot of preprints and brochures [2, 5, 6, 11, 12, 13, 14, 15, 21, 22, 23, 26, 36, 43, 50, 52, 56, 57, 59, 60, 62, 69, 70].

Some Cyrillic versions of TEX and some Cyrillic extensions of the CM font family have been created, including:

1. Protvino package (authors: S. Klimenko, B. Malyshev, A. Samarin, et al., [2, 5, 11, 12, 13, 14, 15, 21, 22, 23, 26, 43, 56, 57]);
2. *CyrTUG* package 1992 (authors: O. Lapko, A. Khodulev, I. Makhovaya [25]);
3. *CyrTUG*–emTEX package 1994 (authors: O. Lapko, A. Khodulev, I. Makhovaya, S. Strelkov [33, 51]);
4. A. Shen package [59];
5. NCC–LATEX package (author A. Rozhenko [53, 62]);
6. ViTEX package (authors: M. Bronstein, M. Vinogradov) [10, 66]);
7. VTEX package (authors: N. Kornev, M. Vulis [68]);
8. wTEX/wLATEX (authors: Yu. Ivanov, V. Korenkov, A. Raportirenko et al.).

Certainly there have also been quite a number of Cyrillic versions which have not made it into the mainstream or enjoyed popular usage. This is to be expected, since TEX is a public domain package. Almost all of the Cyrillic versions are based on the emTEX package as it is one of the most available high-quality programs.

A similar situation has occurred in the development of Cyrillic fonts. Among the best known creators of Cyrillic fonts, we can cite:

1. B. Beeton;
2. T. Ridgeway;
3. N. Glonty;
4. O. Lapko, A. Khodulev ("LH");
5. M. Bronstein, M. Vinogradov;
6. A. Shen;
7. N. Kornev, M. Vulis;

Most of the fonts are Cyrillic extensions of the Computer Modern family. Having several versions and quite a number of fonts that differ from each other on ϵ underlined the need for a common version that combined the best features of each.

There have been numerous attempts along this line. The Cyrillic TEX Users Group gave the problem of standardization top priority. The *CyrTUG*–emTEX package was widely used and was included in the NTG's CD-ROM, 4All TEX, as well as being made available via CTAN.

At a seminar entitled "Nauchno-TEXnicheskie sredy" (held at the MechMath faculty, MSU, 1995–1996), led by E. Pankratiev, *CyrTUG* President, the main issue is the question of standardization. A. Rozhenko has taken the lead in e-mail discussion about the topic; and the Russian Foundation for Basic Research has also taken steps in that direction. Some of the results of this activity are presented at our meeting now.

Because of differing tastes, reaching an agreement on this matter will be difficult. But because we wish to resolve this matter, we meet and join not only in languages groups but in one big family of different nationalities. There is safety in numbers.

References

[1] A. V. Astrelin, "Graphics for TEX: A new implementation", Proceedings of the 9th European TEX Conference, Arnhem, September, 1995, pp. 1–3.

[2] L. S. Bazhanova, "Vvedenie v TEX" (Introduction to TEX), IHEP Preprint 90–116, Protvino, 1990.

[3] B. Beeton, "Mathematical symbols and Cyrillic fonts ready for distribution." *TUGboat* 6,2 (1985), pages 59 – 63.

[4] B. Beeton, "Mathematical symbols and Cyrillic fonts ready for distribution (revised)." *TUGboat* 6,3 (1985), pages 124 – 128.

[5] L. V. Beliaevskaia and I. A. Gritzaenko, "Maloe opisanie TEX'a" (A small description of TEX), IHEP Preprint, Protvino, 1991.

[6] A. S. Berdnikov and S. B. Turtia, *TEX and Drawing* (*TEX i grafika*), Parts 1, 2, Politekhnika, Saint-Petersburg, 1995, ISBN 5–7325–0385–4.

[7] A. S. Berdnikov and S. B. Turtia, "TEX plotter — a program for creating 2D and 3D pictures", Proceedings of the 9th European TEX Conference, Arnhem, September, 1995, pp. 5–9.

[8] A. S. Berdnikov and S. B. Turtia, "VFComb—a program for design of virtual fonts", Proceedings of the 9th European TEX Conference, Arnhem, September, 1995, pp. 11–16.

[9] M. A. Evgrafov and L. M. Evgrafov, *TEX. Rukovodstvo po naboru matematicheskikh tekstov* (*TEX. The textbook on typsetting of math texts*), Moscow, Fiziko-Matematicheskaia Literatura VO Nauka, 1993, p. 80, ISBN 5-02-015116-5.

[10] V. E. Figurnov, *IBM PC dlia pol'zovatelia* (*IBM PC for user*), 4th edition, Financy i statistika, Moscow, 1994, pp. 175–179.

[11] N. M. Fomina, N. L. Glonty, and I. A. Gritzaenko, *Katalog TEX shriftov v IHEP* (*Catalogue

of TEX Fonts Families in the IHEP), IHEP Protvino, 1991, p. 73.

[12] N. L. Glonty, I. A. Gritzaenko, et al., *TEX v IHEP: 5. Shrifty i rabota s nimi* (*TEX in the IHEP: 5. Fonts and how to work at*), IHEP, Preprint 92–127, Protvino, 1992.

[13] N. L. Glonty, I. A. Gritzaenko, et al., "Kratkoe opisanie driverov" (A short description of drivers), *ProTeX*, 1992, p. 15.

[14] N. L. Glonty, I. A. Gritzaenko, et al., "Terminy i opredeleniia" (Terms and definitions), *ProTeX*, 1992, p. 4.

[15] N. L. Glonty, S. V. Klimenko, et al., "Metaproekt Kirillovskogo alfavita dlia pechataiuschikh ustroistv c vysokim razresheniem" (Metaproect of Cyrillic alphabet for printers with high resolution), IHEP, Preprint 90–66, Protvino, 1990.

[16] M. Goossens, CyrTUG'94, Sept. 7–11, TTN, **3** (1994), No 4.

[17] M. Goossens, "Goossens at EuroTEX'94 in Gdansk", *MAPS*, **13** (1994), No 2.

[18] M. Goossens, F. Mittelbach, and A. Samarin, *The LATEX Companion*, Addison-Wesley, 1994, p. 528, ISBN 0-201-54199-8.

[19] I. V. Gorbunova, "Russian-Speaking User: From Chi-Writer and Ventura Publisher to TEX; Learning Difficulties." *TUGboat* 14, (1993), pages 333 – 334. [TUG'93 Conference Proceedings.]

[20] I. A. Gritzaenko, S. V. Klimenko, "TEX — komputernaya systema avtorskoy podgotovki publikacii" (TEX—a computer system for typesetting by author), *Monitor-Aspect*, 1993, No 1.

[21] I. A. Gritzaenko, S. V. Klimenko, et al., "LATEX. Obschaia kharakteristika deistvuiuschei versii" (LATEX. General characteristic of the present version), *ProTeX*, 1992, p. 27.

[22] I. A. Gritzaenko, S. V. Klimenko, V. K. Malyshev, and A. V. Samarin, "TEX v IHEP: 1. Obschaia kharakteristika" (TEX in the IHEP: 1. General characteristic), IHEP Preprint 91–54, Protvino, 1991.

[23] I. A. Gritzaenko, V. K. Malyshev, and A. V. Samarin, "TEX v IHEP: 2. TEX i LATEX na VAX'akh" (TEX in the IHEP: 2. TEX and LATEX on the VAX's), IHEP Preprint 91–55, Protvino, 1991.

[24] T. Jurriens, "TEXniques in Siberia", *NTG bijeenkonst*, **7** (1991) No 2.

[25] A. B. Khoddulev, I. A. Makhovaya, "On TEX experience in Mir Publishers", Proceedings of the 7th European TEX Conference, Prague, September, 1992, pp. 37–43.

[26] S. V. Klimenko, A. V. Samarin, "K voprosu o mobilnosti elektronnoi documentatsii" (About portability of electronic documentation), *Programmirovanie*, No 5, 1989, pp. 80–91.

[27] D. E. Knuth, *Vsio pro TEX* (*The TEXBook*), Russian translation by M. Lisina, editors S. Klimenko, S. Sokolov, Izdatelstvo AO RDTEX, Protvino, 1993, p. 575, ISBN 5-900614-01-8.

[28] N. A. Kornev, *U sistemy TEX poiavilsia "vectornyi" rodstvennik VTEX* (*Now TEX has a "vector" relative VTEX*), READ.ME, 1994, No 1.

[29] K. van der Laan, 6th European TEX Conference, *NTG bijeenkonst*, **7** (1991) No 2.

[30] K. van der Laan, 7th European TEX Conference, *MAPS*, **9** (1992) No 2.

[31] K. van der Laan, "CyrTUG'93 and some more", *MAPS*, **11** (1993) No 2.

[32] K. van der Laan, "CyrTUG'94 and some more", *MAPS*, **13** (1994) No 2.

[33] O. Lapko, "Makefont as a part of *CyrTUG*-emTEX package", Proceedings of the 8th European TEX Conference, Gdańsk, September, 1994, pp. 110–114.

[34] O. Lapko, I. Makhovaya, "A Russian style for Babel: problems and solutions." *Proceedings of the Ninth European TEX Conference* (Sept. 4–8, 1995, Papendaal, Netherlands). Ed. Wietse Dol, pages 289 – 293. [The original Russian version of this paper appeared in *TUGboat* 16, (1995), pages 366 – 372, with a revised English version entitled "The Style `russianb` for Babel: Problems and solutions," on pages 364 – 366.]

[35] S. Lesenko, "T1part: Printing TEX Documents with Partial Type 1 Fonts." *TUGboat* 16,3 (1995), pages 265 – 268. [TUG'95 Conference Proceedings.]

[36] M. V. Lisina, "Plain TEX. Osnovnye poniatiia i katalog kommand" (Plain TEX. The basics and catalog of commands), IHEP Preprint 95–58, Protvino, 1995, p. 156.

[37] S. M. Lvovskiı, *Nabor i verstka v pakete LATEX* (*Typsetting and layout in LATEX*), Kosmosinform, 1994, p. 327, ISBN 5-900242-11-0.

[38] S. M. Lvovskiı, *Nabor i verstka v pakete LATEX. 2-e izdanie* (*Typsetting and layout in LATEX. 2nd edition*), Kosmosinform, 1995, p. 373, ISBN 5-900242-17-x.

[39] V. K. Malyshev, "BaKoMa Fonts Collection", *TTN*, **3** (1994), No 4.

[40] V. K. Malyshev, "Problems of the conversion of METAFONT fonts to PostScript Type 1." *TUGboat*, 15, 3 (1994), page 200 (abstract only). [TUG94 Conference Proceedings.]

[41] V. K. Malyshev, "Problems of the conversion of METAFONT fonts to PostScript Type1." *TUGboat*, 16,1 (1995), pages 60–68.

[42] V. K. Malyshev, M. Goossens, "Partial Font Embedding Utilities for PostScript Type 1 Fonts." *TUGboat*, 16,1 (1995), pages 69–77.

[43] V. K. Malyshev, A. V. Samarin, "TEX v IHEP: 3. TEX na IBM PC" (TEX in the IHEP: 3. TEX on IBM PC), IHEP Preprint 92–15, Protvino, 1992, p. 40.

[44] V. K. Malyshev, A. V. Samarin, D. Vulis, "Russian TEX", *Les Cahiers GUTenberg* (1991), No 10–11, Proceedings of the 6th European Conference and GUTenberg'91.

[45] V. K. Malyshev, A. V. Samarin, "TEX Integrated Shell for IBM PC", *Les Cahiers GUTenberg* (1991), No 10–11, Proceedings of the 6th European Conference and GUTenberg'91.

[46] I. Makhovaya, "CyrTUG", *TTN*, **1** (1992), No 1.

[47] I. Makhovaya, "News about CyrTUG and Russian TEX users", *TTN*, **2** (1993), No 1.

[48] I. Makhovaya, "Knuth visits St. Petersburg", *TTN*, **3** (1994), No 3.

[49] I. Makhovaya, "Donald Knuth — pochetnyi doktor Sankt-Peterburgskogo Universiteta" (Donald Knuth — Doctor of honorary of St. Petersburg University), *Tekhnologiya programmirovaniya*, **1** (1995).

[50] E. Nurminskii, *Tablitsy? Zaprosto. Metodicheskie rekomendatsii* (*Tables? It is easy*). Institute of Applied Mathematics, Vladivostok, 1995.

[51] H. Partl, E. Shlegl, and I. Hyna, *TEX. Kratkoe rukovodstvo* (*A short textbook*). Russian translation with additions by R. Zagretdinov, editor I. Makhovaya, A part of *CyrTUG*-emTEX package, Moscow, 1994, p. 41.

[52] G. M. Petrova and V. M. Rudenko, "TEX dlia nachinaiuschikh" ("TEX for the beginners"), IPM RSN Preprint No. 511, Moscow, 1992, pp. 56.

[53] A. I. Rozhenko, *Rukovodstvo po rabote v sisteme NCC-LATEX. Chast 1. Opisanie Sistemy. Chast 2. Prilozheniia* (*Textbook on NCC-LATEX. Part 1. Description of NCC-LATEX. Part 2. Appendicses*), VC SO RAN Publishers, Novosibirsk, 1993, p. 352.

[54] "Russian books about TEX", *TTN*, **3** (1994), No 4.

[55] "Russians Visit TUG Headquarters." *TUGboat* 11,2 (1990), page 313.

[56] A. V. Samarin, "TEX v IHEP: 4. Ob'edinenie teksta i grafiki" (TEX in the IHEP: 4. The join of texts and pictures), IHEP Preprint 92–48, Protvino, 1992, p. 24.

[57] A. V. Samarin, "Vvedenie v LATEX" (Introduction to LATEX), IHEP Preprint 90–110, Protvino, 1990, p. 115.

[58] A. Samarin, A. Urvantsev, "CyrTUG, le monde TEX en cyrillique", *Les Cahiers GUTenberg*, No 12 (1991).

[59] A. Shen, *LATEX. Kratkoe opisanie* (*LATEX. A short descripton*), Moscow, Nezavisimyi Moskovskii Universitet, 1993, p. 85.

[60] B. M. Shirokov, *PCTEX. Prostoi sposob iziaschnogo oformleniia matematicheskikh textov: uchebnoe posobie* (*PCTEX. A simple way of elegant decoring of math texts: Textbook*), Izdatelstvo Petrozavodskogo Universiteta, Petrozavodsk, 1994, p. 80. ISBN 5-230-08971-7.

[61] A. Slepukhin, "A package for Church–Slavonic typesetting." *Proceedings of the Ninth European TEX Conference* (Sept. 4–8, 1995, Papendaal, Netherlands). Ed. Wietse Dol, pages 331–337. [The original Russian version of this paper appeared in *TUGboat* 16, (1995), pages 376–379, with a revised English version on pages 373–376.]

[62] V. V. Smirnov, "Realizatsiia kirillicheskogo alfavita v sisteme podgotovki dokumentov LATEX" (A realization of Cyrillic alphabet in LATEX), Preprint SA USSR, Sib. branch. 955, Novosibirsk, 1991.

[63] M. D. Spivak, *Voskhititelnyi TEX* (*The Joy of TEX. 2nd edition*), Russian translation by I. Makhovaya, editor A. Khodulev, Mir Publishers, Moscow, 1993, p. 285.

[64] K. O. Telnikov and P. Z. Chebotaev, *LATEX. Izdatelskaia systema dlia vsekh* (*LATEX. The publishing system for everybody*), Novosibirsk, Sibirskii Khronograf, 1994, p. 294. ISBN 5–87550–032–8

[65] M. Vinogradov, "Pro TEX i nemnogo pro drugikh" (About TEX and about others a little), *Mir PK* (1992) No 1.

[66] M. Vinogradov, "Russian TEX: New eight bit font and IBM PC equipment", Proceedings of the 7th European TEX Conference, Prague, September, 1992, pp. 149–153.

[67] D. Vulis, "Notes on Russian TEX." *TUGboat* 10,3 (1989), pages 332–336.

[68] M. Vulis, *Vectornyi TEX i ego primenenie* (*Vector TEX and its application*). Saint-Petersburg, PiC, 1994.

[69] R. V. Zagretdinov et al., *Izdatelskaia sistema LATEX. Kratkoe rukovodstvo* (*Publishing system LATEX. Short textbook*), Kazan State University, Kazan, 1994, p. 96.

[70] A. I. Zhurov and I. I. Karpov, "Osnovy TEX'a" (Basic TEX), IPM RSN Preprint No. 518, Moscow, 1992, p. 54.

◇ Irina A. Makhovaya
Mir Publishers
2, Pervyi Rizhskii pereulok
Moscow, 129820, Russia
`irina@mir.msk.su`

Graphics

LaTeX, *dvips*, EPS and the web ...

Sebastian Rahtz

Abstract

Browsers of TeX question *fora* like `comp.text.tex` will often be asked what are the issues surrounding Encapsulated PostScript, and how one goes about making EPS files from LaTeX output, and maybe using them on the World Wide Web. This short note offers some suggestions.

1 What and why is EPS?

EPS stands for Encapsulated PostScript; EPS files *are* PostScript, but they conform to a minimum standard of good behaviour. This is so they can be included in other documents, possibly resized or rotated. In practice EPS means not using certain commands which have global effects (don't worry, this is quite rare), and inserting structured comments (starting with `%%`) which tell other programs something about the file. The *PostScript Language Reference Manual* goes into great depth describing what these comments can contain, but the minumum that is necessary for practical purposes are:

1. A first line starting `%!PS-Adobe`; *dvips*, for instance, puts `%!PS-Adobe-2.0 EPSF-2.0` in its output, meaning that it claims conformance with version 2 of the EPS standard (we are now at version 3);

2. A 'BoundingBox', like `%%BoundingBox: 33 101 584 715` which tells applications how much space on the page is occupied.

How do you turn PS files into EPS files? They probably are already, if they come from a reputable bit of software (avoid anything from Micro$oft) — a good check is to see if there is a BoundingBox.

You will come across three types of problem with files that look like EPS. Firstly, the BoundingBox may not be accurate; since this determines how much space will be left in enclosing applications like TeX, it matters. Keith Reckdahl's recent tutorial in *TUGboat* goes into detail on this problem.

Secondly, your file may be *serious* EPS, and use all the facilities of structured comments to specify what sort of resources (fonts etc) it expects you to supply when you deal with it. This is bad news if you are in TeX world outside a Macintosh. Look out for lines with words like `ProcSetsNeeded`.

Thirdly, your file may think it is EPS, but in fact breaks the rules, and has weird PostScript in it. The rescue technique is to read it with a forgiving PostScript interpreter, and get a new version written out. Three programs to try are:

1. Adobe Acrobat Distiller; this turns PostScript files into PDF, and Acrobat Exchange can then load them, and save them as ordinary PostScript. Since it is written by Adobe, Distiller is an extremely powerful PostScript interpreter, and can cope with almost anything you throw at it. It is not cheap, but worth having.

2. Recent versions of Adobe Illustrator share some of the Acrobat code, and can read PostScript files, as well as edit PDF files.

3. The free GhostScript is now a very mature and sophisticated product. It understands all of the current Level 2 PostScript, and can turn it onto a wide variety of bitmap forms. Version 4 (released in June 1996) also performs many of the functions of Distiller, and it already reads PDF files and writes PostScript. Unfortunately, its handling of PostScript text to PDF is at present unfinished. However, you can still use GhostScript to read your PostScript and write it out again as a bitmap (eg TIFF).

2 What about dvi to Encapsulated PostScript?

Most TeX systems, free or commercial, supply a dvi to PostScript driver; most of them write out more or less acceptable Encapsulated PostScript, but three are especially well-featured (in the author's experience): the Macintosh Textures driver, Y&Y's *dvipsone* for DOS and the free *dvips*. Since the latter is available for all platforms, is well-supported, and is probably the finest of its type,[1] we shall concentrate on that.

If you want to produce re-useable PostScript output from *dvips* (and this includes output destined for Acrobat Distiller), the absolute priority is to use outline fonts, not the PK fonts traditionally used by TeX. You can either use traditional fonts (usually commercial, like Adobe Times, but GhostScript now comes with an excellent free set donated by URW) or Computer Modern itself in PostScript Type 1 format. Either buy these from Y&Y for Windows and Unix or Blue Sky for Macintosh, or use Basil Malyshev's BaKoMa set, of almost comparable quality.[2]

If you do not use outline fonts, and re-use your output scaled up, you will not like the effect of Figure 1 at all, compared to Figure 2. If you want to turn your documents into PDF, Distiller will produce vile results from PK fonts.

The second priority is to get the right bounding box. Surprisingly many applications cheat by simply making it the page size, regardless of whether the whole area is used. *dvips* does this by default too, but has a command-line option `-E`, which asks it to try and calculate the actual extent used. Note that EPS files are, by definition, only one page, so you also have to use *dvips* options to select just one page. There are two caveats when preparing the input. Firstly, make sure you do not include a page number (try `\pagestyle{empty}` in LaTeX), or else the bounding box will cover that too. Secondly, *dvips* does not always work out the extent of text correctly. For instance, if you wrote (why, I have no idea):

```
Hello\raisebox{10pt}[0pt][0pt]{Up there}!
```

you would be asking LaTeX to raise *Up there* off the baseline, but to pretend that it has no effect on the height calculation. *dvips* will believe this, and calculate a bounding box on the *claimed* height. If you use complicated add-in packages like PSTricks, which add in arbitrary PostScript code, you will also end up in real trouble. In these cases you can either adjust the BoundingBox by hand, or place invisible marks in LaTeX to make sure that *dvips* recognizes the full extent.

A useful trick to remember if you think that TeX knows what you want, but *dvips* does not, is to make judicious use of color. Suppose you wanted to use PSTricks to encircle a mathematical symbol, you might write:

```
absurd \pscirclebox{$\surd$}
```

TeX leaves the right space, since the PSTricks macros understand what is going on, but *dvips* is told to draw the circle in raw PostScript, and the bounding box calculation ignores that. The result is that the limits are set just around the size of the letters. If we wrote:

```
\framebox{absurd \pscirclebox{$\surd$}}
```

it would work correctly, because *dvips* would look at the enclosing frame, not just the words. But you end up with an unwanted box; so make it (in effect) invisible by writing:

```
{\color{white}\fboxsep{0pt}%
 \framebox{%
  {\color{black}absurd
  \pscirclebox{$\surd$}}%
  }%
}
```

This creates a white frame around black text; LaTeX proceeds happily, and so does *dvips*, calculating the right extents, but nothing shows on paper. Obviously, this only works in a monochrome environment.

3 LaTeX to EPS to GIF to Web

Why do we do all this in practice? Often, these days, because people want their LaTeX mathematical output on the World Wide Web, and their only recourse is to embed GIF images in their HTML. The sophisticated *latex2html* program does all this for you; its technique is worth understanding, as it has general utility; the sequence of events is:

1. Place bits of LaTeX in an special file, one fragment per page, and with no page numbers;
2. Run LaTeX to generate a multi-page dvi file;
3. Use *dvips*' `-i` and `-S` options to generate one self-contained output file per page;
4. Give each page to GhostScript, and ask it to render them in pbm (Portable Bitmap) form;

[1] For several years, *dvipsone* has offered partial downloading of fonts, a very powerful feature, but this is now coming into *dvips*; there are also flaws in *dvips*' use of structured EPS comments, and Textures is superior in this respect.

[2] Windows-worshippers may prefer to get into the world of TrueType fonts, which are available for Computer Modern from Kinch Computer Company.

$$-\Phi_0 \frac{\partial}{\partial\varphi}(\Phi_1 a\sin\varphi) - \Phi_1 \frac{\partial}{\partial\varphi}(\Phi_0 a\sin\varphi) - A_1 \left[\Phi_0 + \frac{\partial}{\partial a}(a\Phi_0)\right] \sin\varphi = -a\Phi_0 f \sin\varphi \tag{1}$$

Figure 1: Bitmap EPS file, enlarged and distorted

$$-\Phi_0 \frac{\partial}{\partial\varphi}(\Phi_1 a\sin\varphi) - \Phi_1 \frac{\partial}{\partial\varphi}(\Phi_0 a\sin\varphi) - A_1 \left[\Phi_0 + \frac{\partial}{\partial a}(a\Phi_0)\right] \sin\varphi = -a\Phi_0 f \sin\varphi \tag{1}$$

Figure 2: Outline font EPS file, enlarged and distorted

5. Use the PBMplus/Netpbm utility *pnmcrop* to trim away white space;
6. Use the *ppmtogif* utility to convert the result to a GIF image.

Note that it does *not* use the `-E` option for *dvips*, but relies on simply removing all white pixels until just text is left. This has the advantage that it avoids the problem we saw in the last section, but it has three disadvantages:

1. The PBM utilities are primarily Unix tools, and many people do not have access to them;
2. The cropping process is memory-intensive, slow and eats temporary disk space;
3. The cropping forces everything to the baseline, effectively. A character like em-dash (—) which sits above the baseline, will be cropped above and below, so that the placed GIF looks wrong.

The core of the problem is the use of GhostScript, which always creates a page-sized bitmap, even if there is only one word on the page. What we want is for GhostScript to render just the portion of the image inside the bounding box, if we *do* use the `-E` flag for *dvips*. We can achieve this by giving GhostScript a customized page size, which is the size of the bounding box. Then we can insert some extra PostScript code to move the image so that it starts at the 0,0 coordinate (adjusting the bounding box accordingly). GhostScript then displays or converts the image just within the desired area, and no cropping is needed.

The transformations of the bounding box can be achieved using *epsffit*, which is part of Angus Duggan's `psutils` collection (CTAN:`support/psutils`); the page size change is most easily done using a Level 2 PostScript operator `setpagedevice`. Thus a PostScript file which starts:

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 135 528 284 668
...
```

needs to be transformed to something like:

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 149 140
<< /PageSize [149 140] >> setpagedevice
gsave -135 -528 translate
...
grestore
```

Here we have worked out the width and height of the enclosing rectangle (149×140 units), moved the origin down to 0,0 on the page, and set the page size. PostScript purists will shudder at the `setpagedevice` command, and point out that this is probably illegal in Encapsulated PostScript, but as long as we only use this file strictly in the controlled environment of GhostScript, we are safe enough. Figure 3 lists a simple Perl script which performs the necessary changes to a PostScript file for GhostScript to eat, without any need for *epsffit*[3]

Now that GhostScript is only rendering the desired area, we can use its builtin bitmap output facilities. The Unix or DOS command line:

```
gs -dNOPAUSE  -q -r100 -sDEVICE=tiffg4 \
 -sOutputFile=foo.tif foo.ps -c quit
```

[3] I am aware that it does not cope with an `(atend)` bounding box...

```
#!/usr/local/bin/perl
$bbneeded=1;
$bbpatt="[0-9\.\-]";
while (<>) {
 if ( /%%BoundingBox:(\s$bbpatt+)\s($bbpatt+)\s($bbpatt+)\s($bbpatt+)/ )
 {
     if ($bbneeded) {
         $width = $3 - $1;
         $height = $4 - $2;
         $xoffset = 0 - $1;
         $yoffset = 0 - $2;
         print "%%BoundingBox: 0 0 $width $height\n";
         print "<< /PageSize [$width $height] >> setpagedevice\n";
         print "gsave $xoffset $yoffset translate\n";
         $bbneeded=0;
    }
}
else {  print; }
}
print "grestore\n";
};
```

Figure 3: A Perl script to transform an EPS file for GhostScript

will generate a TIFF fax group 4 image (GhostScript does not support GIF output directly, for legal reasons) at 100dpi of just the imaged area of the PostScript file `foo.ps` with no further ado. GhostScript version 4 adds anti-aliasing facilities; using the Netpbm tools under Unix, we can create a variant GIF image, using the command line:

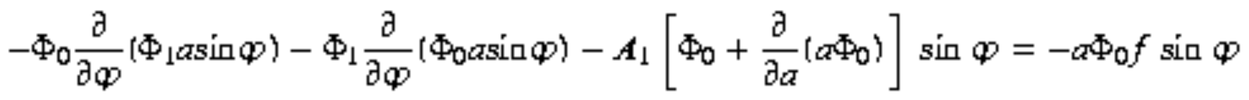
```
 gs -r100 -dNOPAUSE  -q -sOutputFile=-  \
  -sDEVICE=pnm -dTextAlphaBits=4 \
  -dGraphicsAlphaBits=4 foo.ps -c quit | \
  ppmtogif -interlace \
  -transparent \#ffffff > \
  equation.gif
```

Figures 4 and 5 show the result of transformations with and without anti-aliasing. There is one remaining problem — the World Wide Web browsers can usually align images top, middle or bottom; but what if we have an image of some characters with descenders below the base line? Bottom alignment of the images places the bottom of the descenders on the baseline; top alignment is riduculous, and middle alignment is not quite right either. The answer is to use middle alignment, and make TEX lie to *dvips* (and thence down the chain) about the extent of the character; making its depth equal to its height, and then middle aligning it in the Web browser, has the desired effect. So how do we make TEX lie? Here is my suggestion:

```
\newsavebox{\@Fragment}
\def\Fragment#1{%
 \savebox{\@Fragment}{#1}%
 \@tempdima\ht\@Fragment
 \@tempdimb\dp\@Fragment
 \ifdim\@tempdima>\@tempdimb
  \dp\@Fragment\@tempdima
 \else
  \ht\@Fragment\@tempdimb
 \fi
 \fboxsep0pt
 \color{white}%
 \fbox{%
 {\color{black}%
  \box\@Fragment}%
 }%
}
```

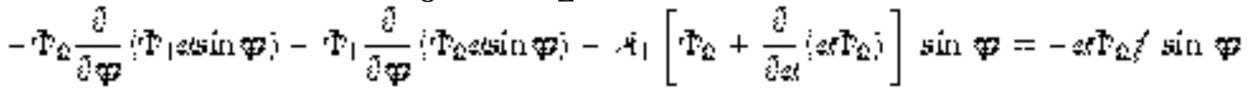
I use the LATEX box framing command to ensure that *dvips* thinks the depth is there, with the same color trick as we saw earlier.

Unfortunately, there is a side effect — an HTML browser loading the resulting GIF image mid-aligns the image and sticks the 'ballast' white space into the line below, making an unsightly gap (see Figure 6, where the Greek etas have a small descender). With the current browser technology, there is little than be done about this. In practice, we will have to check first whether there *is* any descender; if so, we use the mid-align technique, and accept the gap;



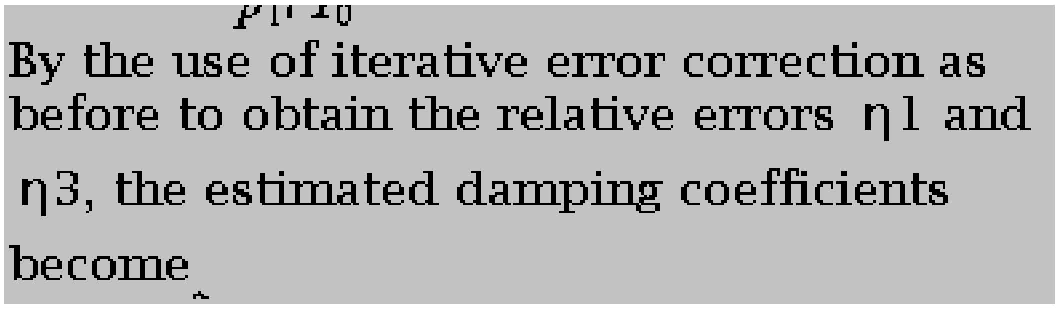
$$-\Phi_0\frac{\partial}{\partial\varphi}(\Phi_1 a\sin\varphi) - \Phi_1\frac{\partial}{\partial\varphi}(\Phi_0 a\sin\varphi) - A_1\left[\Phi_0 + \frac{\partial}{\partial a}(a\Phi_0)\right]\sin\varphi = -a\Phi_0 f\sin\varphi$$

Figure 4: LaTeX → dvi → EPS → GIF



$$-\Phi_0\frac{\partial}{\partial\varphi}(\Phi_1 a\sin\varphi) - \Phi_1\frac{\partial}{\partial\varphi}(\Phi_0 a\sin\varphi) - A_1\left[\Phi_0 + \frac{\partial}{\partial a}(a\Phi_0)\right]\sin\varphi = -a\Phi_0 f\sin\varphi$$

Figure 5: LaTeX → dvi → EPS → GIF, anti-aliased



By the use of iterative error correction as before to obtain the relative errors η1 and η3, the estimated damping coefficients become

Figure 6: Mid-aligned GIF image in Netscape

if there is not, we can make a simpler process and use bottom alignment.

It is imperative, of course, that Web-making readers do not take these examples as 'recipes', without both a precise specification of the desired Web page, or an understanding of some of the basic image-processing techniques. The aim here has simply been to show how relatively trivial and efficient it is to create bitmap output from LaTeX and *dvips* using the free facilities of GhostScript.

◇ Sebastian Rahtz
Elsevier Science Ltd
The Boulevard
Langford Lane
Kidlington
Oxford
UK
`s.rahtz@elsevier.co.uk`

Graphics and TeX — A Reappraisal of METAFONT/MetaPost/PostScript

Kees van der Laan

Abstract

It is all about the author's first steps in METAFONT, to create graphics for inclusion in TeX documents, with a wink to MetaPost and ...PostScript. The graphics comprises graphs of math functions, 2-D pictures and 2.5-D images of 3-D objects via projection techniques; 4-D for varying viewing angles is touched upon. Some highlights on macro writing in METAFONT have been selected, and the appendix provides a list of examples I have collected.

Introduction

The handling of graphics in TeX scripts has a long history.[1] There are three approaches from the document preparation point of view:

- TeX alone
- TeX and METAFONT
- use of third-party graphics tools

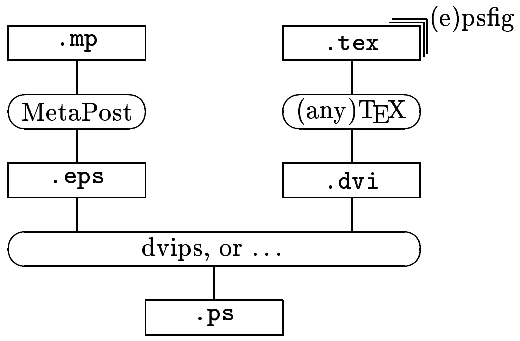
Almost everybody uses encapsulated PostScript (`epsf` for short) as the medium for merging graphics with TeX scripts, in order to get the results out.[2]

TeX alone. LaTeX's picture environment is the most common example for this class, although plain TeXies might use the macros from `gkpmac`,[3] a subset of LaTeX's picture functionality for use with plain TeX. An option I have developed within BLUe's format system introduces the use of 'turtle graphics,'[4] which was used in my "Publishing with TeX" (PWT) user's guide for simple fractals.[5] Interesting too is Gurari's approach (1994).

In scientific circles one problem is how to paste up mathematical graphs electronically. One approach is to calculate the graphs via Pascal, for example, letting it generate the TeX code for the graph. A few years ago I shuffled and typeset bridge hands via this method (van der Laan 1990). Now, however, I would solve the first problem via PostScript straightaway, and I would shuffle the cards, etc. via TeX alone.

Another problem deals with integrating text and graphics. For TeX alone integrating text with simple drawings is no problem at all; this is for me a reason to watch out continuously for what can be done via TeX alone. However, PostScript allows text to be integrated with graphics; better still, text is also considered as graphics, with the extra advantages of scaling or rotating.

Example *(Text integrated with drawings)*



The above graphic illustrates the process for combining TeX and MetaPost.[6]

TeX and METAFONT. John Hobby has recognized the power of METAFONT for designing (systematic) graphics and has married PostScript's outlines to METAFONT to arrive at his MetaPost program, banning the bitmap approach.[7] This is the path I am on, to emulate Naum Gabo's constructive art.[8]

Other approaches currently available include the `mftoeps` package by Jackowski and friends, which transforms METAFONT files into PostScript and vice versa; and `mfpic` from Leathrum and Tobin, which applies METAFONT's character handling technique to export graphics in general.

However, the main subject of this paper will be graphics via METAFONT.

Use of third-party tools. Of late more and more sophisticated graphics and multi-media software

[1] In the old mainframe days, documents were prepared with spaces left for graphics and tables, prepared by other tools, to be pasted up.

[2] Alas, there is no standard as yet for the use of `\special`-s. I hope Rokicki (1995) succeeds with his proposed standard.

[3] Used for typesetting *Concrete Mathematics* (Graham, Knuth, Pastashnik 1989).

[4] See my "Turtle Graphics and TeX — a child can do it," elsewhere in these proceedings.

[5] In this paper the Hilbert curves of order 1 and 2 have been handled via turtle graphics.

[6] Courtesy John Hobby.

[7] Designed under UNIX but also ported to DOS, Macintosh, ...AT&T has released MetaPost and add-ons in the public domain. Thank you.

[8] An artist; born Pevsner in Brjansk, 1890, died 1977 in the United States.

has been emerging, allowing *interactive* graphics (as opposed to systematic, reproducable, declarative graphics), among other things. Happily, the import and export of PostScript files is possible and therefore software, such as Adobe Illustrator, Photoshop, CorelDRAW, etc., can cooperate with TEX and METAFONT. Of course one could use PostScript throughout.

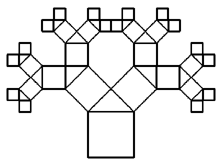
All of the approaches have their pros and cons. What to use — and when — depends as usual on your circumstances. However, third-party tools will not be dealt with in this paper.

METAFONT

For me, learning METAFONT was easier than learning TEX. I picked up the flavor from Knuth's first book in the field, *TEX and METAFONT: New Directions in Typesetting* (1979). Next, I read *The METAFONTbook*, to absorb the ideas, possibilities and details. Finally, and inevitably, I practised with graphics examples borrowed from the literature.[9] A superb summary of the language and what you can achieve with it is given in Hobby (1992). As well, there are introductory documents such as Tobin's 'METAFONT for Beginners' (1993) and Jackowski's GUST tutorial (alas, only available in Polish).

As I built up my list of graphics (see Appendix), I found the path data structure an eye-opener. It shed new light on algorithms for the drawing of Hilbert curves, Sierpiński curves, Pythagorean trees[10] and the like, which are usually formulated recursively.

Example *(Pythagorean tree)*



The use of the path data structure in combination with METAFONT operations on pictures (e.g. `addto`), yields elegant, concise and fast non-recursive programs.[11]

[9] This has resulted in files which I can easily walk through on my Mac, using Blue Sky's public domain METAFONT; it is not (yet) a database to load selectively from.

[10] Done in TEX with turtle graphics from my BLUe's format.

[11] Wirth (1976) has discussed the trade-off between data structures and algorithms. Apparently, he had not thought of the path data structure at the time.

MetaPost and extensions. MetaPost is (nearly) upwards compatible with METAFONT, and concentrates on graphics. It has turned away from the bitmap approach and combines the goodies of PostScript with METAFONT. MetaPost also provides for the integration of text and graphics, next to suitable I/O.

Hobby's graph extension (1993) is all about typesetting scientific graphs — the functionality of troff's grap recast in MetaPost (from Hobby's Introduction):

- automatic scaling
- automatic generation and labeling of tick marks or grid lines
- multiple coordinate systems
- linear and logarithmic scales
- separate data files
- ability to handle numbers outside the usual range
- arbitrary plotting symbols
- drawing, filling and labeling commands for graphs

Why?

I have a keen interest in the work of Naum Gabo. When I first heard of METAFONT it occurred to me that I could emulate his works. To put it in another way. I was curious whether METAFONT could also be used conveniently as a *design* tool for 3-D objects. From a computer science point of view, Gabo's constructive sculptures are very interesting, especially those composed of regular surfaces.[12] I supposed that if Gabo were alive today, he would have exploited the use of computers, since programming a computer — professionally known as software engineering — is a constructivistic activity.

PostScript straightaway. PostScript is a de facto standard and can be included in (LA)TEX documents. For the moment PostScript is intermediate

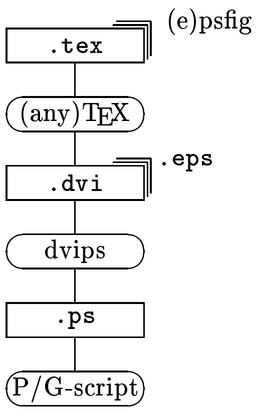
[12] A regular surface is determined by its boundary of which points are connected by straight lines. It conveys a picture of a 3-D object, via 1-D information.

in the chain dvi→ps→pdf;[13] a nodding knowledge of PostScript can only be beneficial.

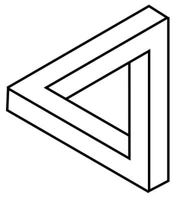
If we consider MetaPost to be PostScript with METAFONT as the user interface, then using MetaPost will bring you sooner or later to PostScript. After METAFONT, learning just a little bit of PostScript only cost me a couple of weeks. Known teasers — typesetting along curved paths, symmetric drawings, accurate drawing of math functions — these can be done elegantly by PostScript alone, and then included at the `.dvi` level. The problem of rotating document elements in general requires more interaction between (LA)TEX and PostScript.[14]

One interesting thing: because of METAFONT's path data structure, I uncovered a new coding for the Hilbert and similar curves, along with a systematic de-recursion technique.

Example *(Process flow using only TEX and PostScript)*



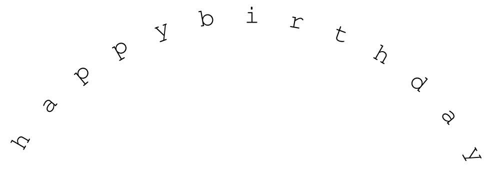
Example *(Escher's impossible triangle)*



```
%!PS EPSF
%%Title: Escher's impossible triangle
%%Creator: cgl (inspired  by Guy Shaw)
%%CreationDate: 1996.05.23:1858
%%BoundingBox: -40 -40 40 40
%%Pages: 1
%%EndProlog
%%Page: 1 1
%100 500 translate
3{25  34 moveto
  25 -34 lineto
  17 -38.2 lineto
  17 20 lineto
 -17.6 0 lineto
120 rotate
}repeat
stroke
showpage
```

Remark. I have also written a variant of the above, with only the first point specified and the rest obtained via (symmetry) operations. Although much more elegant I suspect it less intelligible, and I have therefore suppressed that code here.

Example *(Text set along an arc)*



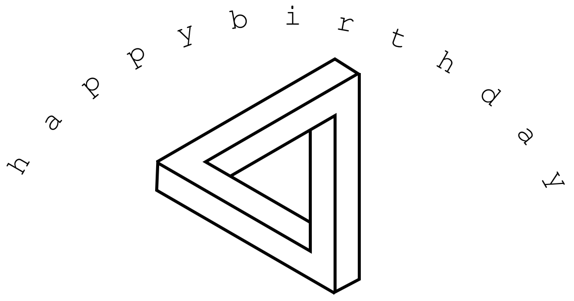
```
%!PS EPSF
%%Title: Typesetting along arcs
%%Creator: cgl
%%CreationDate: June 4 1996
%%BoundingBox: -100 50 100 125
%%Pages: 1
%%EndProlog
%%Page: 1 1
/Courier findfont 10 scalefont setfont
%150 650 translate
%
/text (happybirthday) def
60 rotate
0 1 12{0 100 moveto
  text exch 1 getinterval show
  -10 rotate
}for
stroke
showpage
```

Remark. Joseph Romanovsky (personal communication) has suggested that `kshow` — which allows kerning and positioning to be under the user's control — is the PostScript operator to typeset a string in such an unintended way.

[13] Adobe's Portable Document Format, which abstracts from the preparation tools and concentrates on the user side — the consumer — by the concise `pdf` format and the Acrobat reader, distiller (cross-referencing), exchange and ... multi-media tools.

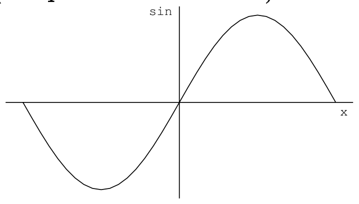
[14] There exists `rotatebox.tex`.

Example *(Seal)*



```
%!!PS EPSF
%%Title: Typesetting along arcs: Seal
%%Creator: cgl
%%CreationDate: June 6 1996
%%BoundingBox: -100 10 100 110
%%Pages: 1
%%EndProlog
%%Page: 1 1
/Courier findfont 10 scalefont setfont
150 650 translate
%
/text (happybirthday) def
gsave
60 rotate
0 1 12{0 100 moveto
  text exch 1 getinterval show
  -10 rotate
}for
stroke
grestore
0 50 translate
3{25  34 moveto
  25 -34 lineto
  17 -38.2 lineto
  17 20 lineto
 -17.6 0 lineto
120 rotate
}repeat
stroke
showpage
```

Example *(Graph of sine function)*



```
%!PS EPSF
%%Title: Sine function
%%Creator: cgl (inspired  by Batagelj)
%%CreationDate: May 27 1996
%%BoundingBox: -200 -110  200 110
%%EndProlog
/Courier findfont 10 scalefont setfont
%figure scaled by 100
%x-axes
-200 0 moveto 200 0 lineto
%label
-5 -10 rmoveto (x) show
%y-axes
0 -110 moveto 0  110 lineto
%label
-25 -5 rmoveto (sin) show
%function
-180 0 moveto
-180 10 180{%from step to
   dup sin 100 mul%(x, 100sin x)
   lineto} for
stroke
showpage
```

The inclusion of `.eps` files can be done via `(e)psfig.tex` in cooperation with dvips.

Examples from my METAFONT Anthology

The various examples of codes outlined below were processed on a Mac, using Blue Sky's public domain METAFONT. How to code the pictures, unblurred by the `shipit` details, was the purpose.

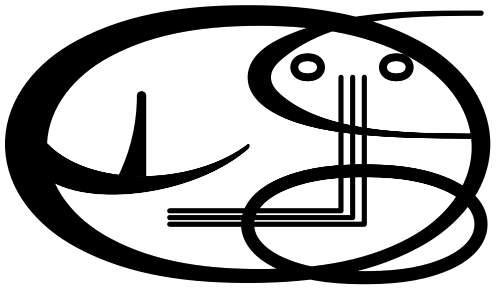
When the METAFONT shipping out of characters is used, the code must be modified: enclosing it with `beginchar` and `endchar` and providing `beginchar` with the appropriate arguments (as treated in *The* METAFONT*book*).

To use MetaPost, a few adaptations are needed: e.g., deleting the bitmap operations `cullit`, `screenstrokes`, and so on, and enclosing the picture with `beginfig` and `endfig` (or `begingraph` and `endgraph`, when the graph extension is used). The effects of reverse video via

```
addto blackbackground also
            -blackpicture
```

has to be adapted too; for example, via the use of (white) color.

Cat. This example is all about the use of a variable width pen. It gives an impression of what can be attained by METAFONT/MetaPost with respect to classical drawing.[15]



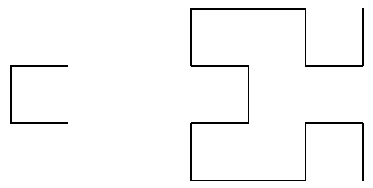
[15] I also developed a version still using a variable width pen but where but `fill` and `unfill`, more suitable for PostScript, were used.

METAFONT code The code is too lengthy (≈50 lines) to be included here. A MetaPost version will soon be available.

PostScript code The MetaPost code yielded a PostScriptfile — much bigger, alas — which is also available, and can be postprocessed if one wishes.

Hilbert curve. In Wirth (1976), the drawing of Hilbert and Sierpiński curves has been treated as essentially recursive. Via the path data structure and copying and rotating of paths built so far, it can be nicely de-recursified.

A Hilbert curve consists of 4 (rotated) copies of a base element connected by 3 straight lines, the 3 edges of a square. H_0 is a dot. The base element of H_k — a Hilbert curve of order k — is H_{k-1}, $k = 1, 2, \ldots$ H_1 and H_2 have been drawn below.[16]



The above graphic results from the following code:

```
$$\unitlength5ex
  \hbox{\qquad
  \vbox to3\unitlength
  {\offinterlineskip
   \vss\W1\S1\E1\vss\vss}
  \kern25ex
  \vbox to3\unitlength
  {\offinterlineskip
     \S1\W1\N1%rotated H_1
     \W1      %connector
     \W1\S1\E1%H_1
     \S1      %connector
     \W1\S1\E1%H_1
     \E1      %connector
     \N1\E1\S1%rotated H_1
    \vss}
}$$
```

TeX code A general TeX macro — parameterized over the order — reads as follows, where `\global<wind>` does what you expect:

```
\def\hlb{\ifnum\orderh=0 \blh\fi
   {\advance\orderh-1
   {\trsfst\hlb}\globalE1
   \hlb\globalN1
   \hlb\globalW1
   {\trssec\hlb}}\relax}
\def\trsfst{\let\exch\globalE
 \let\globalE\globalN
 \let\globalN\exch
    \let\exch\globalW
 \let\globalW\globalS
 \let\globalS\exch}
\def\trssec{\let\exch\globalE
 \let\globalE\globalS
 \let\globalS\exch
    \let\exch\globalN
 \let\globalN\globalW
 \let\globalW\exch}
\def\blh#1\relax{\fi}
```

METAFONT code

```
%Hilbert curve, variant.
%METAfont experiments, code builds upon
%Wirth's A+DS=P, pp130-133 (more concise,
%  uses path data structure; no redoing
%  of already constructed paths.)
%The code has been adapted to build up
%the *path*, and is non-recursive.
%December 1995, cgl@rc.service.rug.nl.
tracingstats:=1;
proofing:=1;screenstrokes;
autorounding:=0;
pickup pencircle scaled 1;
def openit = openwindow currentwindow
from origin to (screen_rows,screen_cols)
at (-20s,15s)enddef;
%
path p; s=10;
sz:=0;p:=origin;%H_0 size and path
n=5;            %Order of H-curve
for k=1 upto n: %H_1,...H_n
p:= p transformed (identity rotated 90
         reflectedabout (origin,up))--
    p shifted ((-sz-1)*s,0)--
    p shifted ((-sz-1)*s,(-sz-1)*s)--
    p transformed (identity rotated -90
             reflectedabout (origin,up)
            shifted (-sz*s,(-2sz-1)*s));
sz:=2sz+1;
draw p shifted(0,-10sz-10); showit;
endfor
end
```

Remarks. Order 4 overflows the rounding table limit (300) with the default autorounding. The connecting straight lines are implicit via `--`. Note the building up of the paths in `p`.

PostScript code Romanovsky has transliterated my (recursive) METAFONT code[17] into a concise PostScript program. A rewrite in the spirit of my TeX macro reads as follows:

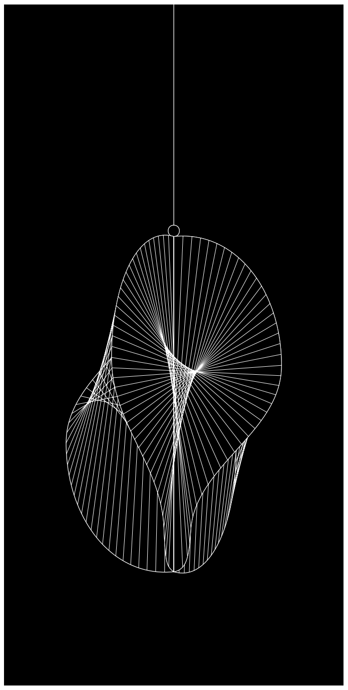
```
%reflect about (origin--(1,-1))
/R{90 rotate -1 1 scale}def
/invR{-1 1 scale -90 rotate}def
%reflect about (origin--(1,1))
/M{-1 1 scale 90 rotate}def
/invM{-90 rotate -1 1 scale}def
%size of element
/scgl 9 def
%
```

[16] Done in TeX, using turtle graphics from BLUe's format.

[17] Not provided here but will be included in my `anthology.mf`.

```
/H{dup 0 gt
   {1 sub
    M H invM scgl 0 rlineto
    H 0 scgl rlineto
    H scgl neg 0 rlineto
    R H invR
    1 add}
   if
}def
```

Linear construction in space II. Thirty years ago I was captivated by Gabo's constructive art, especially his 'Linear construction in space'-like objects. From the METAFONT point of view the following example is all about how to handle a 3-D object — that is, how to describe, project and draw Gabo's 3-D constructive art:



How to do this? How to develop a general technique?

Design First of all, it was necessary to view the object from different angles, which entailed the use of projection techniques.[18]

While programming the object in METAFONT, I had to resolve several issues:

a. how to transform a curve in space
b. how to preserve shape under projection of (a discretisation of) the curve while joining the projected points by METAFONT's splines
c. how to create equidistant points along a curve
d. how to avoid blurring lines
e. how to emulate the used (perspex) material

Coding I solved b and c by first creating the basic shape of a boundary in 2-D, and then replacing the curve by a set of (nearly) equidistant points along the curve. The coding of c reads essentially as follows, where p10 takes over from p1:[19]

```
p10:=for t:=1 upto 19:
     point .05t of p1..endfor origin;
```

The rotation of a boundary curve (for example, from the *yz*-plane into the *xz*-plane) is simply coded as follows, where p100 takes over from p10, etc.:

```
%in yz-plane (the screen)
p100:= for k=0 upto n-1:
  pointtopair(0,xpart(point k of p10),
              ypart(point k of p10))..
    endfor pointtopair(0,0,0);
%in xz-plane
p200:= for k=0 upto n-1:
  pointtopair(xpart(point k of p10), 0,
            ypart(point k of p10))..
    endfor pointtopair(0,0,0);
```

The projection is done via `pointtopair`, which in its simplest version reads as follows:

```
def pointtopair(expr x,y,z)=
%Purpose: The projection of 3D point
%into a pair in the projected plane.
%Arguments: x,y,z coordinates
   (-.6x+.8y,-4/13x-3/13y+12/13z)
enddef;
```

Some of the regular surfaces blur the picture.[20] To clear this up I removed the hidden line parts: to erase what is hidden is determined by the boundary of what is in front:[21]

```
erase fill p100..reverse p200..cycle;
```

The regular surface which is full-blown in sight, has been drawn simply via:

```
for k=0 step 1 until n:
draw point k of p100..point n-k of p200;
endfor
```

To emulate the 'light' caused by the perspex material, I used reverse video, as explained in *The*

[18] See Lauwerier (1987) for an introduction. It contains many examples in ... BASIC!

[19] Note that paths of dynamical length, determined at runtime, are created. I intend to fine-tune this by creating truly equidistant points along a curve via the use of METAFONT's `solve` and Hobby's `arclength`.

[20] In reality Gabo's aim was that from every viewpoint the object could be seen completely. There are no hidden lines in his art. He achieved this by using perspex and nylon.

[21] Knuth uses `overdraw` (1986, p. 243, ex. 13.11), which is very nice. His watchband logo has not been made of perspex apparently.

METAFONT*book*(pp. 115; 118 for the 'dangerous bend').

MF code Too lengthy (≈100 lines) to be included here. The MetaPost program will soon be available. Consult CTAN.

PostScript code As before, the MetaPost code yielded a much larger PostScriptfile; it too is available and can be postprocessed if one wishes.

Macro Facilities

Macro writing in METAFONT is completely different from macro writing in TeX. This paper is not intended as a tutorial on macros; it only provides a few highlights. An appetizer.

FIFO. My favorite FIFO paradigm — first-in-first-out — is implicit in the (var)def parameter handling. For example, the `max` macro (Appendix D of *The* METAFONT*book*, pp. 290–191) allows as argument a list of undetermined length:

```
max(a)   max(a,b,c)
```

Remark. A variable number of arguments is common in METAFONT; for example, `definepixels` and such can be invoked similarly. This is a consequence of (the abstract) `text` as parameter 'type.' MetaPost's `buildcycle` macro makes use of this feature, too, by allowing a list of paths, of undetermined length, as argument. Very useful.

Generic macros. For the `max` macro, for example, the type of the arguments can be numeric, pair or string. This is possible because METAFONT allows for testing for the type of an argument. This generic feature — the same macro for all relevant types — is powerful.

Neat, this abstraction of type and number of arguments, and definitely in agreement with Knuth's aim:

The rules are intended to work the way you expect them.[22]

Gobbling. Another unusual feature is the infix primary `gobbled`, which not only absorbs the argument *after* but also *before.*[23] Infix operators can be defined with primary, secondary or tertiary levels of precedence.

Clipping boundary. Clipping is not provided as such by METAFONT. However, with `cullit` and `cull`, a picture can be clipped. Below, the current picture is confined to a (scaled) square, and provided with a fret:

```
...%picture so far
%reduce all pixels to 0 or 1
cullit;
%make pixel value of picture 2
%within the square
fill unitsquare scaled 100;
%retain picture within the square
cull currentpicture keeping (2,2);
%draw the boundary
draw unitsquare scaled 100;
```

The clipping by a square boundary is just an example to convey the idea. The approach can be applied to all kinds of shapes: for example, to a ring, as done by Jackowski in his EuroTeX95 paper (1995). Note that PostScript and MetaPost do have the concept of clipping path.

Length of a curve. The macro `length` can be applied to a path, resulting in the maximum 'time' rather than the arc length. Hobby applied Simpson's quadrature rule, which yields the following concise approximation (because we know the formula of the Bezier spline):[24]

```
vardef lengthpath expr p=
save dz; pair dz[];
dz0=point 1 of p - point 0 of p;
dz1=point 2 of p - point 1 of p;
dz2=point 3 of p - point 2 of p;
.5(length(dz0)+length(dz0+2dz1+dz2)+
   length(dz2))
enddef;
```

$$\int_0^3 \|B'\|\,dt \approx .5(\|\Delta z_0\| + \|\Delta z_0 + 2\Delta z_1 + \Delta z_2\| + \|\Delta z_2\|)$$

If the path's name is `p` an invoke might read `lengthpath p`, or for a subpath `lengthpath(subpath (3,6) of p)`.

Selective loading. In BLUe's format system the mechanism of selective loading has been used to build a database of tools, pictures and so on. This functionality can also be implemented in METAFONT. The file to load selectively from consists again of triples: list element tag, a symbolic token,

[22] Some codes in Appendix D are not that intuitive, however.

[23] Peruse Appendix B of *The* METAFONT*book* for these kinds of features.

[24] Note that the approximation is only good for sufficiently smooth curves. Split up complex curves in simple ones. Another approach is mentioned by Gibbons (1995). The length of a Bezier spline is bounded by its convex hulls; the smaller the piece, the closer the upper and lower bounds. Repeated division of the curve and summing the lengths of the pieces yields the length.

and text enclosed by parentheses and ended by a semicolon. The list element tag macro has 2 arguments: the implicit suffix — which is compared as string with the name of what we want to select — and the text enclosed by parentheses. If the suffix agrees with the required name, a macro of this name is defined, with the text as replacement text. For example:

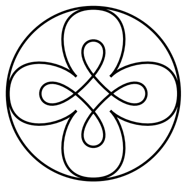
```
vardef lst@#(text t)=
  if str @#= s:%s=selection key
    def @#= t enddef
  fi enddef;
input macro.lst
```

The (toy) file `macro.lst` might read:

```
lst na (draw unitsquare scaled size;);
lst ns (rt);
```

The result with `s:="na";` reads:

```
def na=draw unitsquare scaled size;
enddef;
```



Pitfalls in Compatibility

For my cat, the METAFONT program processed by MetaPost does not yield the correct result. The moustache has disappeared!?! However, if the moustache code is moved to the end, then the correct result is obtained. What has happened? In my opinion this is a consequence of the fact that pixels have values (*The* METAFONT*book*, p. 109).

> Pixels aren't simply "on" or "off" when METAFONT is working on a picture; they can be "doubly on" or "triply off". Each pixel contains a small *integer* value ...

MetaPost's path is apparently just on or off. A `draw <path>` followed by an overlapping `fill` and `unfill` makes that the path disappears. It is better to let the `draw <path>` follow the `fill` and `unfill`.

Example *(Ring with center)*
The compatible way to program this in METAFONT reads essentially as follows:

```
fill fullcircle scaled 10;
unfill fullcircle scaled 7;
drawdot origin;
```

Another incompatibility is in the clipping functionality. In METAFONT this can be done by manipulating pixel values, while MetaPost provides a primitive biased by a clipping contour.

Summarizing my experiences

MetaPost combines the best of both worlds: METAFONT's language features are enriched with contours and `epsf` output.[25] Moreover, it allows access to PostScript's wealth. For creating pictures to be included in (La)TEX or troff documents, MetaPost can be looked upon as PostScript and a METAFONT user interface. Perhaps METAFONT/MetaPost will find their niche in history as convenient tools to describe pictures concisely and at a high level. Via the use of regular surfaces an impression of 4-D can be obtained from 1-D — the contours — information.

What I like about METAFONT. First of all, I like very much the quality, the stability and its being for free. Next, it is available for nearly every platform. Finally, there are the following pleasing details:

- the declarative nature of the language
- the meta-ness and the generic aspects
- the generalization of variable, subscripted variable or record field variable into ⟨tag⟩ ⟨suffix⟩
- just 3 kinds of arguments: expr (independent of type), suffix, and text
- the operator definitions with built-in priorities (`primarydef`, etc.)
- pen, path and picture data structures and operations
- filling and erasing operations
- operations for intersection points
- (nonlinear) interpolation between curves
- various handy 'syntactic sugars'
- rich tracing facilities[26]

If Descartes' Analytic Geometry is still taught today, it might benefit from METAFONT for visualizing.

What I miss in MF. In the list below of missing items, those marked with a plus sign are provided in MetaPost:

+ outlines or `epsf` output (it is all about bitmaps of fonts)
+ mixing of text and pictures[27]
+ general file I/O (writing and reading of pictures)
+ a suitable number range (it is restricted to the half-open interval [1/256/256 – 4096])

[25] It is nearly upwards compatible; `cullit` and other bitmap operations have to be replaced.

[26] Agreed, a necessary evil.

[27] Not to mention Hoenig's typesetting along curved paths (1991).

+ generality of rotating pictures (restricted to a multiple of 90°)
+ various dashed/dotted lines
+ arrows
+ clipping
+ shading and greyscales
+ color
+ invoking TEX
+ making use of PostScript facilities
− triple datatype — point in 3-D as analogue of `pair`
− tree data structure (pointer/handle)

Then there is Hobby's graph extension with its functionalities (as listed earlier), which I have not yet missed, but which I will need for sure when typesetting scientific graphs gracefully.

METAFONT/MetaPost as a production tool. History has it that TEX has been used mainly by scientist with substantial, complex copy, full of mathematics, tables, or graphics, who wish to publish via the electronic networks, via the Internet, not to mention the creative self-publishing world. METAFONT has been used extensively for font production by linguists for non-Latin alphabets.

I don't know of better tools for graphics which are as reliable, portable, ubiquitous, open, completely documented, stable, and cooperative towards other tools. With respect to the last, one can think of the various printer and screen drivers, PostScript and PDF, and the new arrival, HTML — HyperText Markup Language. Moreover, the twins TEX&METAFONT (and descendants) are in the public domain and ported to every platform.

However, one has to learn the systems to know what is under the hood. TEX&METAFONT are not of the push-the-button type tools, like washing machines or cars. Therefore, education is paramount.

The METAFONT/PostScript experts of the Polish TEX User Group GUST have reported that a necessary condition for METAFONT/MetaPost to become a production tool is that `epsf` function as the medium to ease the cooperation with third-party tools. They also have a standing wish for a big METAFONT. As far as I know the company BoP s.c.[28] is the only firm with METAFONT in production.

The future. Maybe we should no longer use paste-up for figures. How about creating hyperlinks to a picture database? If we want to see the picture we can just click and there it is. This is similar to when we like to hear the music when we read about a composer or read the music notation. Whatever these new ways will bring, I for one like the complete document on paper.

[28] Bogusław Jackowski and Piotr Pianowski.

Acknowledgements

Thank you, Joseph Romanovsky, Geoffrey Tobin and John Hobby, for your friendly support. Piet van Oostrum processed 'linear construction no. 2' via MetaPost. The picture came out, for sure, but at the expense of a large PostScript file. Jos Winnink provided the PostScript files via MetaPost and commented on early versions of the paper.

Further references can be found in the file `lit.dat`, on CTAN in `/macros/blu`.

Have fun, and all the best.

References

[1] Gabo, N. *Sixty Years of Constructivism.* Munich: Prestel, 1985.

[2] Gibbons, J. "Dotted and Dashed Lines in METAFONT." *TUGboat* 16,3 (1995), pages 259 – 264.

[3] Graham, R.L., D.E. Knuth, and O. Pastashnik. *Concrete Mathematics.* Reading, Mass.: Addison-Wesley. 1989. [Includes the `gkpmac` macros, available from CTAN.]

[4] Gurari, Eitan M. *TEX and LATEX: Drawing and Literate Programming*. New York: McGraw-Hill, 1994.

[5] Hobby, J.D. *A User's Manual for MetaPost.* Computing Science Technical Report 162. Murray Hill, NJ: AT&T Bell Laboratories, 1992. [Can be obtained from `netlib@research.att.com` via `send 162 from research/cstr`.]

[6] Hobby, J.D. *Drawing Graphs with MetaPost.* Computing Science Technical Report 164. Murray Hill, NJ: AT&T Bell Laboratories, 1993. [Can also be obtained as noted above.]

[7] Hoenig A.J. "When TEX and METAFONT Talk: Typesetting on Curved Paths and Other Special Effects." *TUGboat* 12,3 (1991), pages 554 – 557.

[8] Jackowski, Bogusław. A GUST tutorial on METAFONT. [Currently being translated into English.]

[9] Jackowski, Bogusław. "A METAFONT-EPS Interface." *Proceedings of the Ninth European TEX Conference* (Sept. 4 – 8, 1995, Arnhem, The Netherlands), 257 – 271. [`mftoeps` is available via ftp from `ftp.pg.dga.pl` in the `/pub/TeX/GUST/contrib/MF-PS` subdirectory.]

[10] Knuth D.E. *TEX and* METAFONT*: New Directions in Typesetting*. Part 1: *Mathematical Typography*; Part 2: *TEX: A System for Technical Text*; Part 3: METAFONT*: A System for Alphabet Design*. AMS. Digital Press. (1979).

[11] Knuth, D.E. *The* METAFONT*book*. Reading, Mass.: Addison-Wesley, 1986.

[12] Lauwerier H.A. *Meetkunde met de microcomputer*. Epsilon 8, 1987. [Basic programs are included.]

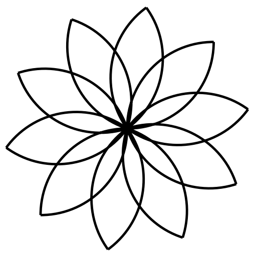
[13] Leathrum, T., and G. Tobin. `mfpic`. 1994. [Available from CTAN in the `graphics` subdirectory.]

[14] Rokicki, T. "A Proposed Standard for Specials." *TUGboat* 16,4 (1995), pages 395–401.

[15] van der Laan, C.G. "Typesetting Bridge via TEX." *TUGboat* 11,2 (1990), pages 265–276.

[16] Tobin, G. METAFONT for Beginners. 1993. [Available from CTAN in the `info` subdirectory.]

[17] Wirth, N. *Algorithms + Data Structures = Programs*. Englewood Cliffs, NJ: Prentice-Hall, 1976.

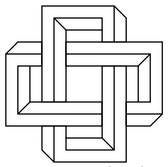


Appendix ToC METAFONT Anthology

I hope that the codings mentioned below will contribute to the METAFONT 'literature.'

```
%Anthology of METAFONT endeavours,
%Started Nov 1995, revision March 1996
%Author/Compositor: Kees van der Laan,
%Hunzeweg 57, 9893PB, Garnwerd, Holland
%                cgl@rc.service.rug.nl.
%The anthology files work under
%BLUe sky's METAFONT (Public Domain)
%on the Macintosh.
%(How to scroll through these examples on
% other systems I don't know.
% Just copy what is of interest for you
% and emulate...
% towards a discipline of coding in
% METAFONT/METAPOST.)
%For other systems copy the example(s)
%you are interested in build a character
%from it and follow the usual procedures
%to include the font in your (La)TeX,
%or use for example the Jackowski/Rycko
%mftoepsf package to produce epsf.
%For use in METAPOST enclose the elements
%by beginfig ...endfig.
%
%Disclaimer borrowed from
%Preface METAFONT book:
%'That's the way how it is with any
% powerful tool:
%    There is always more to learn, and
%    there are always better ways to do
%    what you have done before.'
%
%Table of Contents
%%%Preliminary, file ant.mac
%-Macros used
% (default) openit
% pointtopair projection
% dash
% turtle movements:
%    east, south, west, north
%-Defaults
%%%ToC proper%%%
%
%%%Tiles, file: ant.til
%-Roos and variation via interpath
%-Variations via interpath
%-Mondriaan
%-Alhambra tiles (Courtesy M C Escher)
%-Escher tiles
%-Escher reptiles I
%-Escher reptiles II
%-Escher Buddhas
%-Escher square limit
%-Meeting and meeting apart
%-Moriscs ornament (Courtesy
%                    J V Romanovsky)
%-Chinese porcelain ornament
%       (Courtesy J V Romanovsky)
%-Chinese porcelain ornament II
%-Horak's tiles
%%%2D figures, file: ant.two
%-Cat (varying pen width, non-linear
%     interpolation)
%-Cat II (portable?)
%-Cat III(essential picture)
%-Whirlpool (Courtesy H A Lauwerier)
%-Generalized polygon
% (sides as flexes)
%-Flexes II (Courtesy B Jackowski)
%-Flexes III(Interpath)
%-2D polygon regular patterns
%-Nails (Courtesy J V Romanovsky)
%-Nails II
%-Nails III
%-Yin-Yang
%-Removing Overlap and
% Expanded Stroke
%-Turtle graphics: square spiral
%-Spiral of squares
% (Courtesy H A Lauwerier)
%-Recycle logo (Courtesy P Mackay)
%-Recycle logo II
%
%%%3D, file: ant.thr
%-2.5D simplest example:
% twisted plane
```

```
%-Cube and impossible cube
% (MB 13.7, 2.5D variant)
%-Tetraeder
%-Tetraeder and
% Gabo's torsions inside
%-Octaeder
%-Ikasoeder
%-Triangular edge Moebius band
% (Courtesy Tuckerman)
%-Hobby's pyramid
% (without text etc) in 2.5D
%-Hypercube
%-Spiral 2.5D
%-Emulating Gabo:
% doll's
% Hyperboloid
% Linear construction I
% Linear construction II
% Speric theme
% Speric theme;
%   interpolating surfaces
% Vertical construction 0
%-Toroid of rotating circles
%-Toroid of rotating circles
% (2.5D variant)
%-Cube example,
% varying viewing angles
%-Moebius band,
% varying viewing angles
%
%%%OP art, file ant.opt
%-Optical illusion
% (jiggling squares)
%-Optical illusion (parallel?)
%-Op/kinetical art (Courtesy Soto)
%-Knuth meets Vasarely
%-Vasarely II
%-Vasarely III
%-Vasarely IV
%-Vasarely V
%-Vasarely VI
%-Op art pulsing I
% (Courtesy Jean Larcher)
%-Op art pulsing II
% (Courtesy Jean Larcher)
%-Op art (Profile)
%-Op art (circles and ellipses)
% (Courtesy Schrofer)
%
%%%Fractals, file: ant.frc
%-Cantor dust fractal
%-Hilbert curve
% (Courtesey N Wirth)
%-Sierpinski curve
% (Courtesey N Wirth)
%-W-curve (Courtesey N Wirth)
%-H-fractal
%-Pythagorean tree
%-Pythagorean tree, non-recursive
%-Sierpinski square
% (Courtesy B Jackowski)
%-Sierpinski triangle
% (recursive with use of save)
%-Sierpinski triangle (without save)
%-Sierpinski carpet (via gambling)
%
%%%Math curves
%-Conic sections
% Ellips, hyperbole, parabole
%-Lemniscate
%-Astroid
%-Cardoid
%-(Hypo)cycloid
%-Deltoid
%-Nefroid
%-Spirals: Square, Archimedes,
% Logarithmic, Spheric
%
%%%Various, file ant.var
%-Haralambous' deformations
% of a `circle'
%-Calculation of (arc)length of a
%  Bezier segment via quadrature
%-Exercise from MB: p.176 (solve use)
%-Exercise from MB: 13.8  (star)
%-Exercise from MB: 13.10 (S-figure)
%-Exercise from MB: 13.10 (S-figure
%     contour via `expanded stroke')
%-Exercise from MB: 13.11
% (Moebius band)
%-Exercise from MB: 13.11
% (variant 2.5D Moebius band)
%-Example from MB 14 p134
% (Interpath use: heart interpolation)
%-Exercise from MB: 15.6
% (variant, via paths)
%-Exercise from MB: 20.5
% (variant, parameter sep TeXnique?)
%-n faculty (exercise recursion,
% number range)
%-n asterisks(exercise recursion)
%-quicksort  (exercise recursion)
%%%end ToC
```


Courtesy Woody Baker

◇ Kees van der Laan
Hunzeweg 57
9893 PB Garnwerd
The Netherlands
Email: cgl@rc.service.rug.nl

Book Reviews

Book review: *A TEX Primer for Scientists*, by Sawyer and Krantz

David B. Thompson

Stanley Sawyer and Steven Krantz, *A TEX Primer for Scientists*. CRC Press, Boca Raton, FL, 1993, ISBN 0-8493-7159-7.

In general, I like this book. It is written with the novice user in mind and the presentation is successful. As stated in the preface,

> ...1) It is an aid to the busy scientist, mathematician, or engineer who wants to learn to use the computer typesetting system TEX as quickly and easily as possible; 2) It is a reference for the more experienced TEX user.

The "how-to" approach works. Examples are presented in abundance. This makes the text useful as a copybook. Furthermore, the authors discuss likely typesetting errors and the correct TEX code to use for repairs. When I commit to learning TEX and leave LATEX this will be one of the books I use to effect the transition.

Krantz and Sawyer is organized into two major divisions: *A First Course in TEX* (Chapters 1–7) and *A Second Course in TEX* (Chapters 8–14). The first division focuses on introductory issues, particularly topics required for mathematical typesetting. I believe that a fledgling typesetter should be able to handle most of the tasks required for scientific typesetting after reading the first division. The authors begin their book with the obligatory introductory chapter, then present three chapters on typesetting mathematics. A discussion of text macros and TEX fonts follows in chapter 5. The authors use the final two chapters in the first division to flesh out details required for typesetting technical documents. Items such as type size and margins, headers and footers, references, etc., are discussed. Also included is discussion of the differences between typesetting and word processing: ligatures, hyphens, kerning, and spacing.

Although one might be tempted to consider the first division elementary typesetting, this is not the case. Krantz and Sawyer move beyond elementary typesetting issues and discuss typesetting of complicated formulae. They provide difficult examples and effectively demonstrate production of well-typeset mathematics for the reader. The plentiful examples should provide templates a reader of this book can use right away.

The second division is an investigation of more detailed items. In these chapters are presented the niceties that have kept me using LATEX (but the book is about TEX). Instructions for setting up numbered lists, displays, graphics, and more fonts are presented. Also discussed are low-level programming considerations: variables, counts, boxes, and font magnification. Some additional topics of interest to routine users of TEX are the tabbing environment, table typesetting, and a discussion of typesetting tables of contents, indices, and so forth.

I have only two minor complaints: I found the discussion of hardware and software systems for TEX systems weak. I would have relegated this material to an appendix or deleted it. In addition, I found one minor error (not in the TEX part of the presentation): Krantz and Sawyer mistakenly state that the text editor *QEdit* is part of the MSDOS operating system. This is not true. *QEdit* is the product of SEMWare located in Marietta, Georgia (blatant plug). I use SEMWare's products and can heartily recommend them.

In summary, I recommend *Krantz and Sawyer* as a reference work for new TEX users. I am not an advanced TEX user, so I cannot address the utility of this work for advanced users; however, my subjective assessment is that *Krantz and Sawyer* and *The TEXbook* would be an excellent starting point for moving from novice to hacker.

◇ David B. Thompson
Civil Engineering Department
Texas Tech University
P. O. Box 41023
Lubbock, Texas 79409
`wqdbt@ttacs.ttu.edu`

Book review: *The Advanced TEXbook*, by David Salomon

Włodek Bzyl

David Salomon, *The Advanced TEXbook*. Springer 1995, softbound, xx + 492 pp. ISBN0-387-94556-3. The size is the same as *The TEXbook*..

With the book in my hands I tried to guess what it hides inside. Looking at the title, I expected that it should begin with the repetition of basic material which afterwards would be continued with the detailed exposition of advanced concepts. The table of contents confirmed my guess. Moreover, it is the only book I know which contains two introductions (the first is basic, the second advanced). Introductory material is followed by eighteen chapters where every aspect of the TEX language seems to be explained. The chapters of my immediate interest were:

- Examples of Macros,
- Multipass Jobs,
- Output Routines,
- OTR Techniques,
- Insertions.

The book contains a bibliography, the answers to exercises, and an extensive index.

Browsing through the book revealed that the concepts are usually illustrated by examples and complemented with exercises. Approximately one-third of the book contains material previously published in *TUGboat*. To summarize, the material covered is sufficient to move anyone all the way from the rank of TEXnician to the rank of Grandmaster.

In assessing the accuracy of this work I have attempted to read part of it very carefully. This simple approach discovered many errors and inaccuracies in the presentation. Let us start this part with a citation taken from the book under review.

> Tokens and file I/O are two more features that distinguish TEX from most word processors and page layout programs. They contribute to the power of TEX and should be mastered by anyone aspiring to the title of TEXmaster (or even that of a TEXnician).

Although the first of the two sentences above might sound controversial, the second describes the place to be visited on the way to become the true TEXmaster. Unfortunately tokens could not be mastered with *The Advanced TEXbook*. The author has a wrong idea of the tokenization process. To justify this strong opinion I chose two examples.

Let's start with (p. 172): "We thus cannot say '`\def\endverb{\endverbatim}`' ... because the string in macro `\endverb` starts with $\backslash_0$ instead of $\backslash_{12}$." This is wrong because the replacement text of the `\endverb` macro is the token $\boxed{\texttt{endverbatim}}$.

To learn the concept of token lists in TEX we are told that TEX is a living organism with four key organs: eyes, mouth, stomach, and bowels. Research work of leading TEX anatomist Victor Eijkhout proved that these organs should be given new names (matching the functions they perform): input processor, expansion processor, execution processor, visual processor. David Salomon looking at the anatomical diagram on the page 456 of *The TEXbook* discovered a new organ which he named 'gullet'. Its function is to expand tokens and execute certain commands. It is my opinion that introduction of the new organ which performs some functions of mouth and stomach, and at the same time narrows the functions of the eyes results in a creature which hardly resembles TEX.

In several places the readers are warned by the author: "The macros and the programs listed in this book have been tested by the author but are not guaranteed. They are meant to be read, understood, and modified by the reader for specific applications. They are not meant to be copied and used verbatim." OK, I feel warned.

On page 140 I found two lines of code which should make a comma stick out into the margin, if it occurs next to a line break.

```
\setbox0=\hbox{,} \catcode`,=\active
\def,{,\kern-\wd0\kern\wd0}
```

This piece of code produces an infinite loop. It was not difficult to repair the code.

```
\setbox0=\hbox{,}
\def\comma{,\kern-\wd0\kern\wd0}
\catcode`,=\active
\let,=\comma
```

Next I tried to understand how the macros work. The comma hangs when the line is broken between the two kerns. This agrees with the explanation provided: "If a comma is used at the end of a line, however, the last `\kern` is discarded and the comma is left hanging out on the right." But, on the page 91 we are told: "A line break can only occur at a glue, a penalty, a 'math-off', or a discretionary break." This means that a line break cannot separate kerns. The two kerns cancel each other and the comma does not hang at all. To recapitulate the situation: there are two sentences of which at least one could be true. (By the way, the above code could be criticized on two points more.)

Let me finish the review with personal comments on the design of the text. The design is very stingy on the vertical space. This causes a lot of trouble. Sometimes it looks as if the author added or dropped artificially a line of code to make a page exactly full. The effect is strange. Diagrams are oversized and look ugly. A keyboard and an arrow symbol are used to mark points in the text. Whatever reasonable meaning is attached to them, they are frequently misplaced or put in unnecessarily.

In spite of the above criticism I would like to recommend this book to anyone wanting to pay for the one-fourth of it covering multipass jobs and output routines plus insertions. The reason I offer is simple: these are places where I found a lot of inspiring material.

◇ Włodek Bzyl
Instytut Matematyki,
Gdańsk University,
Wita Stwosza 57,
80-952 Gdańsk,
Poland.
matwb@univ.gda.pl

Tutorials

The New (LaTeX 2_ε) *TUGboat* Macros

Robin Fairbairns

1 Introduction

This note reports on a new set of macros for use by *TUGboat* authors. The macros represent a development of the earlier ltugboat and ltugproc styles that were written for use with LaTeX 2.09. The development was started by others than I, notably Sebastian Rahtz, Michel Goossens, Nico Poppelier and Johannes Braams; I have been assisted in my work both directly and indirectly by many others, including Barbara Beeton, Mimi Burbank and (of course) the LaTeX3 team.

The ultimate aim of the work is to provide a single production environment for the whole of *TUGboat*. That aim has not (yet) been achieved, though it is still believed to be achievable for a significant proportion of papers.[1]

[1] *TUGboat* will continue to accept papers written using the 'plain' macros; it's probably fanciful to hope that *every* paper can be moulded into use within LaTeX.

2 Availability

The macros are released for general use, and are distributed via CTAN[2] in the usual LaTeX way as files `tugboat.dtx` and `tugboat.ins`. When the `.ins` file is processed by LaTeX, the files `ltugboat.cls`, `ltugproc.cls` and `ltugbib.bst` (for ordinary production work) and `ltugcomn.sty` (a cooking pot of useful macros, for documentation, etc.) are produced. The `.dtx` file may itself be processed by LaTeX to produce a formatted (somewhat 'literate') source listing for those who feel they need more detailed description of the macros than is offered in the present note.

3 The general structure of a paper

The author need not understand the details of the production of *TUGboat* — her business is to produce the text that the editor is to mould into the body of an issue.

However, any modern author wants to be able to view the progress of her efforts; for this, she needs to use the *TUGboat* class `ltugboat.cls` — it defines the way an article in a normal issue of *TUGboat* will appear, and is therefore a useful aid.

Similarly, one submitting a paper for presentation at an annual TUG conference may use the `ltugproc.cls`, which will set the paper as it should appear in a proceedings issue of *TUGboat*.

Each paper, therefore, is written as a document that may stand on its own. It starts with a `\documentclass` command, and its body is enclosed in a document environment. Each of the document classes specifies a set of options, described in the next two subsections (3.1 and 3.2). In the ordinary course of events the author needn't bother with any of these options, since the defaults are designed for creating proof copies of papers.

Since the author typesets her paper in a 'proof mode', the paper's published appearance will differ from its appearance at the time of submission. The changes required in this process are the responsibility of the *TUGboat* production team, and the author need not be concerned with them.

3.1 Class options: the ltugboat class

The ltugboat class accepts all the options of the article class (though it suppresses the font-size selection, one/two-side and one/two-column options).

draft Set up for a draft copy of a paper (this is the default setting — the author need not explicitly set it): page numbering to start at 1001, black

[2] From directory `macros/latex/contrib/supported/tugboat`

marks for overfull boxes, and no registration marks.

final Set up for the final copy of a paper: page numbering to come from elsewhere, no black marks, and registration marks to be used.

harvardcite Specify Harvard-style citation; see section 11.

nonumber Sections are not to be numbered; section heading layout is to be as in the 'plain' tugboat styles.

numbersec Sections, subsections and subsubsections are to be numbered (this is the default setting — the author need not explicitly set it).

preprint Set up for a preprint.

rawcite Specify default (numbered) citation; see section 11.

3.2 Class options: the ltugproc class

The ltugproc class accepts all of the options allowed with ltugboat. In addition, it accepts an option that specifies the year of the conference for which the paper is submitted. For example, this year's conference would be specified by option tug96.

The reason that such an option is necessary is that the type design of papers for each year's proceedings issue is decided (in detail) by the editor.

By default, ltugproc class selects the option for this year's conference. The class reads a configuration file `ltugproc.cfg`, in which you may specify the year by a command of the form:

```
\newcommand{\tugProcYear}{97}
```

The class deals with two-digit year selections for conferences up to 2069 (which date was chosen entirely at random, uninfluenced, of course, by the Unix$^{\text{TM}}$Epoch...).

4 Command syntax

We would have liked to offer perfectly uniform syntax for people to use when preparing their papers. Unfortunately, uniform syntax is not available with any widely-available set of macros (though see, for example, the discussions in Ogawa, 1994, Baxter, 1994, and Swift, 1995). In the circumstances, we have sought simply to keep to the spirit of Lamport's (1994), as modified[3] by the LaTeX 2_ε work (see, for example, Rowley, 1994).

In the few cases that it has proved possible to emulate (what seems to a staid old LaTeX programmer, such as the present author) the gay abandon of the syntax of the 'plain' tugboat styles (Whitney and Beeton, 1989), we have done. Nevertheless, on the whole, the new ltugboat macros simply define LaTeX commands and environments, or modify the definitions of LaTeX 'standard' commands. Section 12 lists equivalences between macros defined by the 'plain' package and those defined by the new package.

The 'down' side of this decision is, of course, the 'welter of LaTeX braces' that Barbara Beeton has been heard to complain of in production team discussions. One has to hope that the (near) uniformity of syntax offers those who think like Barbara some small recompense!

[3] The cited edition of Lamport's book documents the modified version of LaTeX, but it's worth emphasising that we use the modified version as the reference.

5 Divisions of the paper

Papers in *TUGboat* may be subdivided in the normal way of a LaTeX article (the classes are defined in terms of LaTeX's article class). Thus the author may use `\section`, `\subsection`, ..., `\paragraph` commands (but `\part` and `\subparagraph` from article are suppressed, and `\chapter`, which doesn't even appear in the parent class, receives the same treatment).

Authors should note that the style of ordinary issues of *TUGboat* makes no distinction between the titles of the divisions; the visual style relies on the section numbers to indicate where the divisions lie in the hierarchy. As a result, the un-numbered '`*`' forms of the `\section`, etc., commands, are inappropriate. The ltugboat class therefore warns the author who (possibly inadvertently) uses one of these forms. (The author who wishes to use un-numbered sections throughout her paper may use the nonumber class option (see 3.1).

By contrast, the conference proceedings issues of *TUGboat* do *not* number their sections, by default. As a result the use of '`*`' forms is equally inappropriate in ltugproc class, and the class equally warns the author. Since the sections don't have numbers, references to section numbers don't work; they therefore (once they're resolved) warn the author of potential problems.

5.1 Abstracts

The classes make provision for abstracts, but the provision is different for the two classes.

The ltugboat class provides two environments, abstract and longabstract. The abstract environment simply typesets its body as an un-numbered section whose title is 'Abstract'. The longabstract environment typesets its body in small text, and separates the abstract from the rest of the paper with a decorative line.

The ltugproc class typesets abstracts as part of the title of the paper, below the names and addresses of the authors, but before the main body of the paper. Therefore, for proceedings articles, the abstract environment must appear in the source of the paper *before* the `\maketitle` command (it may even appear before `\begin{document}`, if the author so chooses).

5.2 Appendices

A paper may have appendices, which are expressed in exactly the same way as they would be in LaTeX article class:

```
\appendix
\section{This is appendix A}
...
\section{This is appendix B}
```

Which will produce 'section' headings similar to:

A This is appendix A

TUGboat articles may have a small extension to this format: this extension was originally developed for proceedings issues, but is also available in normal issues:

```
\begin{appendix}
\section{This is the first one}
...
\end{appendix}
```

Which will produce 'section' headings similar to:

Appendix A This is the first one

In both cases, the subsections are numbered as normal (i.e., as 'A.*n*' in normal *TUGboat* papers, and not at all in proceedings classes).

6 Titles, addresses and so on

The title and author(s) of a paper are quoted using commands that are familiar (in syntax, at least) to most LaTeX users; the `\title` command is exactly that used in the standard LaTeX classes.

The `\author` command is used once for each co-author of the paper, and for each `\author` there should be a `\address` command that gives a (postal) correspondence address. In addition (wherever possible), *TUGboat* likes to quote an email address for authors: for this, the `\netaddress` command is used. Finally, each author may advertise a 'home' Web page, using a `\personalURL` command.

For example, the present paper has at its start:

```
\title{The New (\LaTeXe) \TUB{} Macros}
\author{Robin Fairbairns}
\address{University of Cambridge
    Computer Laboratory\\
  Pembroke Street,\\
  Cambridge, CB2 3QG,\\
  UK}
\netaddress{rf@cl.cam.ac.uk}
\personalURL{http://www.cl.cam.ac.uk/users/...}
\maketitle
```

Since this paper is prepared for a normal issue of *TUGboat* and therefore uses the ltugboat class, the `\maketitle` merely typesets the title of the paper and the author's name. The address(es) are typeset at the end of the paper where the author gives a `\makesignature` command.

The ltugproc class typesets all the information about the author at the head of the paper, when the `\maketitle` is given; this class disallows the `\makesignature` command.

Note that the author had a problem typesetting the above example verbatim: the lines are too long. If the information being given is to be typeset as ordinary text (as in the case of the `\address` line above), it can be 'wrapped' perfectly happily, as in normal text. If one of the verbatim items (`\netaddress` or `\personalURL` commands) is going to be too wide for the column, what is the author to do? (Abbreviating the text, as in the `\personalURL` above, is *not* usually an acceptable option!) Unfortunately, the `%` sign is an entirely acceptable element of both email addresses and URLs, so that the normal 'fall-back' isn't available. Therefore, the classes typeset these electronic addresses in an environment where some of the characters (notably '`.`' and '`/`') are treated as word-divisions for the purposes of laying out the line. This is visible in the case of my personal URL, in the signature block at the end of the present paper.

If the paper is the result of more than one author's labours, a sequence of `\author`, `\address`, `\netaddress` and `\personalURL` commands may be given, as in the following, which comes from a paper given at TUG'95 (slightly edited):

```
\author{Michel Goossens}
\address{CN Division, CERN\\
  ...}
\netaddress{...}

\author{Sebastian Rahtz}
\address{Elsevier Science Ltd\\
  ...}
\netaddress{...}

\author{Robin Fairbairns}
\address{University of Cambridge
    Computer Laboratory\\
  ...}
\netaddress{...}
\personalURL{...}
```

The class files will take care of arranging author names and addresses between the `\maketitle` and (possibly) `\makesignature` commands.

7 Verbatim text

The classes do not at present provide the same wide range of facilities as the 'plain' tugboat style (Whitney and Beeton, 1989); the author had hoped to 'borrow' facilities from a package which is believed to be in development, but in the event that package has not materialised.

For in-line verbatim text, authors should ordinarily employ the facilities of LaTeX itself (the `\verb` macro). This macro, of course, is highly restricted as to its usage (primarily, that it may not appear in the argument of *any* other macro, even `\footnote`).

For 'display verbatim' (to employ the term used by Whitney and Beeton), the classes add a small increment to the functionality of LaTeX's verbatim environment, by introducing an optional argument. The optional argument may contain commands to be executed before starting the verbatim text; the set of commands which have useful effect is strictly limited, but the following are commonly used:

- Font size selection commands: for example, all the display verbatim in the present paper starts with:

  ```
  \begin{verbatim}[\small]
  ```
- The command `\ruled`, which is available *only* in verbatim's optional argument, and specifies that a column-wide rule should be drawn before and after the verbatim text
- One of the `\make*` commands,[4] which change the category code of characters within the verbatim text. This is (of course) a facility that should only be used with the utmost caution, but it can, for example, be employed to provide interesting effects by knowledgeable authors.

Two caveats about the use of this facility should be noted:

- The search for the optional argument can be confused by the appearance of a `[` character as the first of the displayed verbatim. An author who wishes to start verbatim text with a `[` character should provide an empty optional argument (i.e., simply '`[]`') to the verbatim environment.
- The facility is lost when certain packages are loaded. An example is the verbatim package (Schöpf, 1996), which redefines the verbatim environment in its entirety. Of course, any package that loads verbatim (such as moreverb, Duggan et al., 1996) will necessarily have the same effect. (It should be noted that verbatim and moreverb provide some of the facilities that are available in the 'plain' tugboat styles, so that an author could reasonably be tempted to use them. There is no objection in principle to authors using these packages.)

[4] `\makeescape`, `\makebgroup`, . . . , `\makecomment`; used, for example, as `\makeescape\|`

8 Floating inserts

The classes do not make any change to LaTeX's built-in provision for floating inserts, so that authors may generate figures and tables just as they would in any 'normal' LaTeX document. Figure and table captions, and labels referring to them, are also substantially untouched.

However, since both classes typeset in two columns, authors must distinguish between the figure and table environments (which produce floats that are the same width as the column) and the figure* and table* (which produce floats that are the same width as the page).

9 Special-purpose typesetting

The classes define a rather large set of commands for special-purpose typesetting. Some of them are available for historical reasons only, and many are only useful in somewhat restricted circumstances. For this reason, the present paper only outlines a representative, small set of the macros.

9.1 Acronyms and logos

The classes provide macros that produce 'correct' representations of a large number of acronyms and logos; a small representative selection is shown in figure 1.

Macro	*Output*
`\CTAN`	CTAN
`\eTex`	ε-TeX
`\HTML`	HTML
`\ISBN`	ISBN
`\ISSN`	ISSN
`\MF`	METAFONT
`\MP`	METAPOST
`\NTS`	$\mathcal{NTS}$
`\OMEGA`	Ω
`\OTP`	ΩTP
`\SGML`	SGML
`\TUB`	*TUGboat*
`\TUG`	TeX Users Group
`\tug`	TUG

Figure 1: Some of the classes' acronyms and logos

Authors are urged to note the `\acro` command, which is defined in the classes. The visual appearance of (mostly) lower-case English text, with interpolated acronyms in the same point size, is generally unpleasing. Therefore, the `\acro` command typesets its argument slightly smaller than it would otherwise appear: compare 'URL' (`\acro{URL}`, as used above) with 'URL'. Many macros that simply generate calls to `\acro` are defined by the classes; two examples, `\CTAN` and `\tug` of the list in figure 1 have already been used in the present paper.

9.2 Other special typesetting

A small list of special typesetting commands follows: a large set of such commands is defined in the classes, but the list covers most of the 'everyday' ones.

`\cs{`*cmd*`}` Typeset a control sequence name.

`\Dash` Typeset an em-dash, surrounded by thin spaces, only breakable *after* the dash; this is the preferred method of specifying a dash in running text.

`\dash` Typeset an en-dash, in the same way as `\Dash` does.

`\nth{`*n*`}` Typeset an ordinal number. For example, `\nth{1}` is set as 1^{st}, `\nth{27}` is set as 27^{th}, and so on.

`\sfrac{`*num*`}{`*denom*`}` Typeset a fraction to match running text; for example `\sfrac{3}{4}` is set as $^3/_4$.

10 Use of packages

In general, the *TUGboat* team will be sympathetic to authors who wish to use non-standard packages in their papers; indeed, in a journal devoted to the usage of TeX, the editor would be churlish indeed to refuse such usage. However, the team does need to be able to process the paper on the *TUGboat* production computers, and the author who is slapdash about the way she submits her paper will cause confusion and delay. (Remember that the editorial team is a group of volunteers, each working in his or her spare time.)

In general, packages currently on CTAN, and known to work with *current* LaTeX,[5] are unlikely to give problems.

In particular, the team is happy to accept papers using packages that are supported by members of the LaTeX3 team.[6] subject to the two provisos:

[5] The team's version of LaTeX is regularly updated, though not during the course of a production run.

[6] Those in the LaTeX base distribution, or one of those on the `macros/latex/packages` sub-tree on CTAN.

- Use of the verbatim package has implications for the verbatim facilities provided by the classes — see section 7.
- Use of babel almost inevitably implies use of hyphenation patterns that the team may not have installed in their LaTeX format; it is therefore important that the author explains her babel configuration to the editorial team. The minimum documentation required is a copy of the author's `language.dat` file, and copies (or CTAN pointers to) any hyphenation files used.

Usage of other packages should always be subject to negotiation with the team. If the team does not have access to a copy of the package, life is going to be very difficult; authors are urged to be sensible in this regard. A sensible mechanism for submitting out-of-the-ordinary packages (as for paper-specific bibliographies) is by use of the filecontents environment.

TUG has a policy that macro packages described in *TUGboat* should be available for readers to use. Since typing macros from printed sources is such an error-prone undertaking, authors of publicly available packages are urged to submit their macros to the CTAN archives. If a package is only available under restricted terms, authors are urged to make this fact clear when first submitting an article to the editor.

Some facilities are considered inappropriate to delivery by the *TUGboat* classes, and as a result, the *TUGboat* team recommend certain packages to authors.

At present, the list of recommended packages consists of only two, `mflogo.sty` (Vieth, 1995) and `url.sty` (Arseneau, 1996).

Both classes will load the mflogo package if it is present on the author's system; if the package is not present, the classes will emulate its more important features; the package defines METAFONT and METAPOST logos using recent versions of Knuth's `logo10` font family.

The url package is useful when one is typesetting significant numbers of file names, network addresses or URLs; it is being used in the present paper (not least in the bibliography).

11 Bibliography

Bibliographic citations give much grief to the editorial team. Good publishing practice requires that there be editorial control of the way citations in a journal are presented yet, all too often, authors submit articles whose bibliography is formatted according to their preference. The important rules for authors, then, are that they *shouldn't* supply a `.bbl` file (BibTeX processed output), and that they *shouldn't* write out a thebibliography environment, but should rather submit a working `.bib` file with their paper.[7] As with uncommon packages, the filecontents environment is a convenient way to deliver the bibliography file.

A special case is made of the accumulated bibliography of *TUGboat* itself;[8] it is always available to the production team, so that authors may make reference to items from that `.bib` file without further ado.

Two citation styles are supported within *TUGboat* articles, '*raw*' and '*harvard*' (the present article is employing *harvard* citation). The raw citation style uses the 'standard' BibTeX 'plain' (numeric) citation style; its modification by use of Donald Arseneau's cite package[9] is acceptable. Raw citation is selected by default (by execution of class option rawcite).

Harvard citation may be selected by specifying harvardcite as an option of the `\documentclass` command. The macros used derive pretty directly from the *harvard* styles written by Glenn Paulley and now maintained by Peter Williams; the BibTeX style derives from one developed by Patrick Daly.

The basic citation format is 'author-year', but the macros are capable of many variations: this in turn places somewhat of a load on the author to use the correct citation macro. The macros available are shown in figure 2; the figure assumes an entry in the bibliography with authors Tom, Dick, and Harry, and with a 1990 date.

Macro	*Output*
`\cite{key}`	(Tom, Dick, and Harry, 1990)
`\citeA{key}`	(Tom, Dick, and Harry)
`\citeNP{key}`	Tom, Dick, and Harry, 1990
`\citeANP{key}`	Tom, Dick, and Harry
`\citeN{key}`	Tom, Dick, and Harry (1990)
`\shortcite{key}`	(Tom et al., 1990)
	[also has `A` and `NP` variants]
`\citeyear{key}`	(1990)
	[also has an `NP` variant]

Figure 2: The range of citations in harvard style

Plain macro	LaTeX macro
`\head`	`\section`
`\subhead`	`\subsection`
`\subsubhead`	`\subsubsection`
`\list`	itemize, enumerate, etc., environments
`\verbatim`	verbatim or `\verb`
`\figure`	figure or figure* environments

Figure 3: Equivalences between plain and LaTeX macros

12 Equivalences between the 'plain' and the LaTeX packages

A good proportion of the commands in the 'plain' packages also appear (with the same meaning) in the LaTeX classes. Figure 3 gives a brief summary of where the macros differ significantly.

LaTeX itself makes comprehensive provision for lists; the *TUGboat* classes make no attempt to emulate the list facilities of the 'plain' macros.

The 'plain' styles' provision for verbatim text is also somewhat different from the LaTeX approach; the *TUGboat* classes offer a small subset of the extra facilities that the 'plain' styles provide; for more elaborate facilities, the user is referred to the verbatim and moreverb packages (see section 7).

Of course, the syntax of commands given to the LaTeX classes is different (as discussed in section 4); arguments are (almost always) enclosed in braces, and neither of the forms of argument provision promulgated by the 'plain' macros (`\macro`⟨*argument*⟩ `\endmacro` and `\macro *` ⟨*argument*⟩ `*`) are provided by the LaTeX classes.

References

Arseneau, Donald. "The url package". Available from CTAN, `macros/latex/contrib/other/misc/url.sty`, 1996.

Baxter, William Erik. "An object-oriented programming system in TeX". *TUGboat* **15**(3), 331–338, 1994.

Duggan, Angus, R. Schöpf, V. Eijkhout, and R. Fairbairns. "The moreverb package". Available from CTAN, `macros/latex/contrib/supported/moreverb`, 1996.

Lamport, Leslie. *LaTeX, a Document Preparation System*. Addison-Wesley, 2nd edition, 1994.

Ogawa, Arthur. "Object-oriented programming, descriptive markup, and TeX". *TUGboat* **15**(3), 325–330, 1994.

[7] The program bibextract, available from CTAN in directory `biblio/bibtex/utils/bibextract`, provides a convenient mechanism for extracting just the relevant portions of your bibliography database for submitting your paper; many of the common BibTeX-database management packages, such as BibDB or BibTools offer similar facilities.

[8] Available on CTAN as `digests/tugboat/biblio/tugboat.bib`

[9] Available on CTAN as `macros/latex/contrib/supported/cite/cite.sty`

Rowley, Chris. "LaTeX 2_ε update, dateline: 31 January 1994". *TUGboat* **15**(1), 63, 1994.

Schöpf, Rainer. "The verbatim package". Part of the tools bundle, available from CTAN, `macros/latex/packages/tools`, 1996.

Swift, Matt. "Modularity in LaTeX". *TUGboat* **16**(3), 269–275, 1995.

Vieth, Ulrik. "The mflogo package". Available from CTAN, `macros/latex/contrib/supported/mflogo`, 1995.

Whitney, Ron and B. Beeton. "*TUGboat* authors' guide". *TUGboat* **10**(3), 378–385, 1989.

◇ Robin Fairbairns
University of Cambridge Computer Laboratory
Pembroke Street,
Cambridge CB2 3QG,
UK
`rf@cl.cam.ac.uk`
URL: `http://www.cl.cam.ac.uk/users/rf/robin.html`

Using EPS Graphics in LaTeX 2_ε Documents Part 2: Floating figures, boxed figures, captions, and math in figures

Keith Reckdahl

Abstract

This is the second of two papers that explain how to use Encapsulated PostScript (EPS) files in LaTeX 2_ε documents.

The first paper in the series, which appeared in *TUGboat* **17** (1), covered

- the graphics and graphicx packages, which provide commands to insert, scale, and rotate EPS graphics,
- commands which are commonly used in conjunction with EPS graphics,
- use of `dvips` to insert compressed EPS files and non-EPS graphic formats (TIFF, GIF, JPEG, PICT, etc.)
- software for decompression or graphics conversion capabilities, which must be provided by the user.

The present paper covers

- floating figures in various configurations (such as more than one figure in a single float), and the use of the subfigure package,
- creation of boxed figures, by use of the `\fbox` command, or of the facilities of the fancybox package,
- manipulation of the caption of a figure, including use of the caption2 package, and
- modifying the text within an EPS file by using the PSfrag system, for example to include mathematical symbols or equations.

1 The `figure` Environment

Graphics can be inserted as part of a LaTeX `figure` environment, which allows the graphics to float for better formatting, especially for large graphics. The `figure` environment also makes it easy to reference the graphic. The commands

```
\begin{figure}[htb]
  \centering
  \includegraphics[totalheight=2in]{graph.eps}
  \caption{This is an inserted \EPS{} graphic}
  \label{fig:graph}
\end{figure}

The graph in Figure~\ref{fig:graph} is
from an \EPS{} file generated by gnuplot.
```

insert the graphic in a figure and place a caption under the graphic. The optional `\label` command specifies a label which is used by the `\ref` command to reference the figure (the `\label` command must be *after* the `\caption` command). Note that the figure environment can only be used in *outer paragraph mode* and thus cannot be used inside any box (such as `\parbox` or `minipage`).

1.1 Caption Vertical Spacing

While the figure caption is usually placed below the graphic, it can be placed above the graphic simply by placing the `\caption` command before the graphics-inclusion command. For example, the commands

```
\begin{figure}[htb]
  \centering
  \caption{Caption Above Graphic}
  \includegraphics[width=1in]{box.eps}
\end{figure}
```

produce Figure 1.

Figure 1: Caption Above Graphic

Since captions are generally placed below the graphic, LaTeX places more vertical spacing above the caption than below it. As a result, the caption in Figure 1 is placed quite close to the graphic. The spacing above and below the caption is controlled by the two lengths `\abovecaptionskip` (which is `10pt` by default) and `\belowcaptionskip` (which is zero by default). The standard LaTeX commands `\setlength` and `\addtolength` are used to modify these lengths. The commands

```
  \setlength{\abovecaptionskip}{5pt}
  \setlength{\belowcaptionskip}{0.5cm}
```

provides a 5 point spacing above the caption and a 0.5 centimeter spacing below the caption. The commands

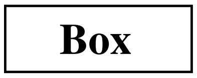
```
  \addtolength{\abovecaptionskip}{5pt}
  \addtolength{\belowcaptionskip}{-5pt}
```

increases the spacing above the caption by 5 points and decreases the spacing below the caption by 5 points. For example, the commands

```
\begin{figure}[htb]
  \setlength{\belowcaptionskip}{10pt}
  \centering
  \caption{Caption Above Graphic}
  \includegraphics[width=1in]{box.eps}
\end{figure}
```

produce Figure 2.

Figure 2: Caption Above Graphic



1.2 Figure Placement Options

LaTeX figures are "floats" whose placement is decided by LaTeX. Since your taste in figure-placement may differ from that of LaTeX, the `figure` environment has placement options

- `h` *Here:* Place the figure in the text where the figure command is located.
- `t` *Top:* Place the figure at the top of a page.
- `b` *Bottom:* Place the figure at the bottom of a page.
- `p` *Page of Floats:* Place the figure on a separate page which contains only floats.

The placement options in the above example are `[htb]` which means that LaTeX first tries to place the figure at that location, then tries to place the figure at the top of a page, and finally tries to place the figure at the bottom of a page. When LaTeX "tries" to place a figure, it checks how many figures are already on the page and other esthetic concerns. If LaTeX determines that the figure wouldn't look good, it tries the next placement option.

The order in which the placement options are specified does *not* make any difference. The placement options are attempted in the order `h-t-b-p` regardless of the order in which the options are specified. Thus `[hb]` and `[bh]` are both attempted as `h-b`.

To make LaTeX "try really hard" in its float placement, put an exclamation point in the placement options (e.g., `\begin{figure}[!ht]`) which makes LaTeX suspend its esthetic rules and do its best to make the requested placement. Even with the `!` option, LaTeX has the final say in the placement and reserves the right to override the request. For example, if the commands

```
\begin{figure}[!ht]
  \includegraphics[totalheight=4in]{graph.eps}
\end{figure}
```

occur 3 inches from the bottom of the page, LaTeX objects to leaving 3 inches of whitespace at the bottom of the page and overrides the `[!h]`, filling the bottom 3 inches of the page with the text which is after the figure in the .tex file.

If you feel LaTeX is making poor float placement decisions, you may need to tweak its placement algorithm by modifying the float parameters (see [1, pages 199-200], [2, pages 141-143], or [3, pages 174-175]).

1.2.1 The float Package's [H] Placement Option

The float package adds an `[H]` option to the `figure` environment which *always* places the float "here". However, this option should normally be avoided, as the `[!ht]` option is a better way of producing the desired behavior.

To use the `[H]` option, include

```
\usepackage{float}
```

in the preamble and put a `\restylefloat{figure}` command *before* the `\begin{figure}[H]` command is used. (See [2, page 149].) When using the `[H]` option, the user is responsible for managing the document to avoid large sections of whitespace.

While the figure environment defined by the float package allows the `[H]` option, it also places the figure caption below the figure environment. While this does not affect simple figures, it prevents captions above graphics as in Figure 1 or the construction of side-by-side and other complex figure arrangements.

2 Landscape Figures

In a document with portrait orientation, there are three methods for producing figures with landscape orientation.

1. The lscape package provides a `landscape` environment, which treats the left edge of the paper as the top of the page, causing any text, tables, or figures in the `landscape` environment to have landscape orientation.
2. The rotating package has a `sidewaysfigure` environment which is similar to the `figure` environment except that the figures have landscape orientation.
3. The rotating package provides a `\rotcaption` command which is like the `\caption` command except that the caption has landscape orientation.

The differences between methods are as follows:

- Both options 1 and 2 place the rotated figure on a separate page. Option 3 produces an individual float which need not be on its own page.
- The full-page figure produced by Option 2 will float to provide better document formatting. Since the figure(s) produced by Option 1 can only float within the landscape pages, this may result in a partially-empty page before the figure.
- The `landscape` environment in Option 1 can be used to produce landscape pages containing any combination of text, tables, and figures. Option 2 produces only rotated figures or tables.

2.1 The `landscape` Environment

The lscape package (part of the standard "graphics bundle" distributed with LaTeX) defines the `landscape` environment; this lets you place landscape pages in a portrait document. The landscape pages are rotated such that the left edge of the portrait page is the top edge of the landscape page.

Entering `\begin{landscape}` generates a `\clearpage` command which prints all unprocessed portrait floats, before switching to landscape orientation. Likewise, `\end{landscape}` prints all unprocessed landscape floats before switching back to portrait orientation.

The entire contents of the `landscape` environment is typeset with landscape orientation. This may include any mixture of text, figures, and tables. If the landscape environment contains only a figure environment

```
\begin{landscape}
  \begin{figure}
    \centering
    \includegraphics[width=4in]{box.eps}
    \caption{Landscape Figure}
  \end{figure}
\end{landscape}
```

the `landscape` environment produces a landscape figure. Note that since the `landscape` environment starts a new page, it may result in a partially-blank page.

2.2 The `sidewaysfigure` Environment

The rotating package provides the `sidewaysfigure` environment which produces figures with landscape orientation. For example

```
\begin{sidewaysfigure}
  \centering
  \includegraphics[width=4in]{box.eps}
  \caption{Sidewaysfigure Figure}
\end{sidewaysfigure}
```

produces Figure 3.

Unlike the `landscape` environment, the figure produced by `sidewaysfigure` can float within the portrait pages to avoid the partially-blank page that the `landscape` environment may produce. However, the `landscape` environment is much more flexible, allowing the landscape pages to consist of a mixture of text, tables, and figures. The rotating package also provides a `sidewaystable` environment for producing tables with landscape orientation. Unlike the `landscape` environments, the `sidewaystable` and `sidewaysfigure` environments cannot contain a mixture of text, figures, and tables.

The default orientation of the figures produced by `sidewaysfigure` depends on whether the document is processed with the `oneside` or `twoside` documentclass option

- When the `oneside` option is chosen, the bottom of the graphic is towards the right edge of the portrait page.
- When the `twoside` option is chosen, the bottom of the graphic is towards the outside edge of the portrait page.

This default behavior can be overridden by options to the `\usepackage{rotating}` command.

```
\usepackage[rotateleft]{rotating}
```

causes the bottom of the `sidewaysfigure` graphics to be towards the left edge of the portrait page (regardless of `oneside` or `twoside` options). Similarly,

```
\usepackage[rotateright]{rotating}
```

causes the bottom of the `sidewaysfigure` graphics to be towards the right edge of the portrait page.

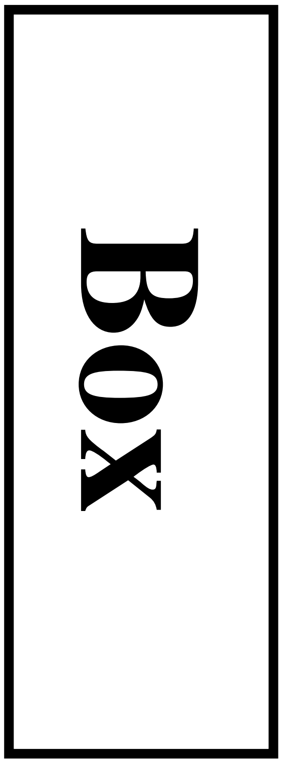


Figure 3: Sidewaysfigure Figure

2.3 The `\rotcaption` command

The methods in Sections 2.1 and 2.2 both produce full-page landscape figures, which may not be necessary for smaller landscape figures. The rotating package's `\rotcaption` command can be used to construct smaller landscape figures. For example

```
\begin{figure}
 \centering
 \begin{minipage}[c]{1in}
  \includegraphics[angle=90,
   width=\textwidth]{box.eps}
 \end{minipage}
 \begin{minipage}[c]{0.5in}
  \rotcaption{Rotcaption Caption}
  \label{fig:rotcaption}
 \end{minipage}
\end{figure}
```

produces Figure 4.

The caption produced by `\rotcaption` is always rotated such that its bottom is towards the right edge of the paper. Unlike the methods in Sections 2.1 and 2.2, the `\rotcaption` command does not rotate the graphics. We therefore added the `angle=90` option in the above example.

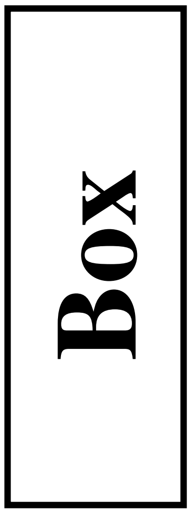


Figure 4: Rotcaption Caption

3 Side-by-Side Graphics

The commands necessary for side-by-side graphics depend on how the user wants the graphics organized. This section covers three common methods of organizing side-by-side graphics

1. The side-by-side graphics are combined into a single figure.

2. The side-by-side graphics each form their own figure (e.g., Figure 12, Figure 13, etc.)

3. The side-by-side graphics each form a subfigure (e.g., Figure 12a, Figure 12b, etc.) of a single figure (Figure 12).

While this section specifically discusses side-by-side graphics, most of the information is also valid for vertically-stacked graphics and complex figures such as Figures 28-34 on Page 297.

3.1 Side-by-Side Graphics in a Single Figure

The two most common methods for placing side-by-side graphics in a figure are

1. Multiple `\includegraphics` commands
2. Multiple `minipage` environments, each of which contains an `\includegraphics` command

3.1.1 Side-by-Side `\includegraphics` Commands

While spacing side-by-side graphics in a figure can be as simple as

```
\begin{figure}
  \centering
  \includegraphics[width=1in]{file1.eps}
  \includegraphics[width=2in]{file2.eps}
  \caption{Two Graphics in One Figure}
\end{figure}
```

there are usually horizontal-spacing commands such as `\hspace{1in}` or `\hfill` between the `\includegraphics` commands. For example,

```
\begin{figure}
 \centering
 \includegraphics[width=1in]{box.eps}%
 \hspace{1in}%
 \includegraphics[width=2in]{box.eps}
 \caption{Two Graphics in One Figure}
\end{figure}
```

produces Figure 5 which is 4 inches wide (1 inch for `file1.eps`, 1 inch for the `\hspace`, and 2 inches for `file2.eps`). This 4-inch-wide figure is centered on the page. If `\hfill` is used instead of `\hspace`, the graphics are pushed to the margins.

3.1.2 Side-by-Side `minipage` Environments

Greater control over the graphics' horizontal and vertical placement can be obtained by placing the commands inside `minipage` environments. For example,

```
\begin{figure}
 \centering
 \begin{minipage}[c]{0.5\textwidth}
  \centering
  \includegraphics[width=1in]{box.eps}
 \end{minipage}%
 \begin{minipage}[c]{0.5\textwidth}
  \centering
  \includegraphics[width=2in]{box.eps}
 \end{minipage}
 \caption{Centers Aligned Vertically}
\end{figure}
```

produces Figure 6.
Some notes on this example:

- Like any other LaTeX object, minipages are positioned such that their baseline is aligned with the current baseline. The minipage `[c]` option defines the minipage's baseline as its centerline. The `[b]` option defines the minipage's baseline as the baseline of the bottom line of the minipage (which is not necessarily the bottom of the minipage). The `[t]` option defines the minipage's baseline as the baseline of the top line of the minipage (which is not necessarily the top of the minipage). See section 4 for information on the `minipage` environment and its placement options.

- The `%` after the first `\end{minipage}` command prevents a space from being inserted between the `minipage` boxes. Such a space would use some horizontal space, preventing both minipages from fitting on the same line.

When the widths of the minipages do not add up to `1.0\textwidth`, the `\hspace` or `\hfill` commands can be used to specify to horizontal spacing. For example,

```
\begin{figure}
 \centering
 \begin{minipage}[c]{1in}
   \centering
   \includegraphics[width=\textwidth]{box.eps}
 \end{minipage}%
 \hspace{1in}%
 \begin{minipage}[c]{2in}
   \centering
   \includegraphics[width=\textwidth]{box.eps}
 \end{minipage}
 \caption{Centers Aligned Vertically}
\end{figure}
```

produces a figure with the same horizontal spacing as Figure 5, but the centers of the boxes are aligned vertically.

3.2 Side-by-Side Figures

In the previous section, multiple `minipage` environments were used inside a `figure` environment to produce a single figure consisting of multiple graphics. Placing `\caption` statements inside the minipages makes the minipages themselves become figures. For example,

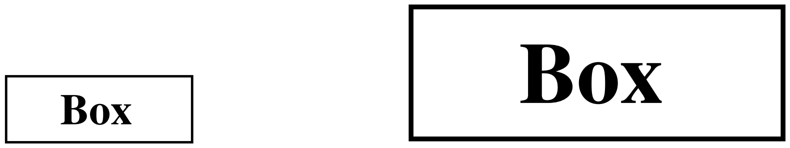


Figure 5: Two Graphics in One Figure

Figure 6: Centers Aligned Vertically

```
\begin{figure}
 \begin{minipage}[b]{0.5\linewidth}
  \centering
  \includegraphics[width=1in]{box.eps}
  \caption{Small Box}  \label{fig:side:a}
 \end{minipage}%
 \begin{minipage}[b]{0.5\linewidth}
  \centering
  \includegraphics[width=1.5in]{box.eps}
  \caption{Big Box}  \label{fig:side:b}
 \end{minipage}
\end{figure}
```

produces Figures 7 and 8.

Although the above commands include *one* figure environment, the commands produce *two* figures. Since the `\caption` command actually produces the figure, figure environments with multiple `\caption` commands produce multiple figures.

3.2.1 Alignment Problems with Side-by-Side Figures

The `[b]` options aligned the bottoms of Figures 7 and 8. However, long captions may affect this alignment. For example,

```
\begin{figure}
 \begin{minipage}[b]{.333\linewidth}
  \centering
  \includegraphics[width=1in]{box.eps}
  \caption{Small Box with a Long Caption}
  \label{fig:side:c}
 \end{minipage}%
 \begin{minipage}[b]{.333\linewidth}
  \centering
  \includegraphics[width=1.5in]{box.eps}
  \caption{Medium Box}  \label{fig:side:d}
 \end{minipage}%
 \begin{minipage}[b]{.333\linewidth}
  \centering
  \includegraphics[width=2.0in]{box.eps}
  \caption{Big Box}  \label{fig:side:e}
 \end{minipage}
\end{figure}
```

produces Figures 9, 10, and 11.
The long caption of Figure 9 means that it is not aligned with the other figures. In this case, the baselines of all the figures are their bottoms, so the alignment can be corrected by changing the `minipage` positioning option from `[b]` to `[t]` which aligns the baselines of the graphics (see Section 4 for information). If the baselines of the graphics do not correspond to their bottoms, the `[t]` option does not produce the desired positioning. Instead, invisible vertical lines (called *struts*) can be placed in the captions of the other figures to make LaTeX think that all the captions are two lines long.

```
\begin{figure}
 \begin{minipage}[b]{.333\linewidth}
  \centering
  \includegraphics[width=1in]{box.eps}
  \caption{Small Box with a Long Caption}
  \label{fig:side:cc}
 \end{minipage}%
 \begin{minipage}[b]{.333\linewidth}
  \centering
  \includegraphics[width=1.5in]{box.eps}
  \caption[Medium Box]
   {Medium Box
      \protect\rule[-\baselineskip]{0pt}
                          {2\baselineskip}}
  \label{fig:side:dd}
 \end{minipage}%
 \begin{minipage}[b]{.333\linewidth}
  \centering
  \includegraphics[width=2.0in]{box.eps}
  \caption[Big Box]
    {Big Box \protect\rule[-\baselineskip]{0pt}
                        {2\baselineskip}}
  \label{fig:side:ee}
```

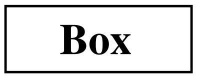


Figure 7: Small Box

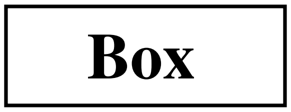


Figure 8: Big Box

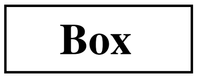


Figure 9: Small Box with a Long Caption

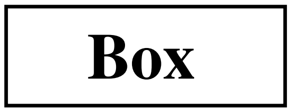


Figure 10: Medium Box

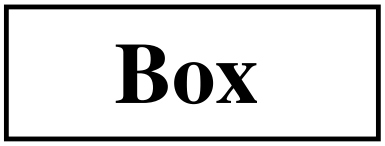


Figure 11: Big Box

```
 \end{minipage}
\end{figure}
```

which produces Figures 12, 13, and 14.

`\rule[`*`start`*`]{`*`width`*`}{`*`height`*`}` produces a vertical line with a width of *`width`* starting *`start`* above the baseline and with a height *`height`*. When the width is zero, the line becomes invisible and is called a *strut*. In the above captions, the strut

```
 \rule[-\baselineskip]{0pt}{2\baselineskip}
```

starts one line below the baseline and continues to the top of the current line. This makes LaTeX think that, like the Figure 12 caption, the captions for Figures 13 and 14 are two lines tall. Since the `\rule` command is fragile, the `\protect` command must be used so that `\rule` can be used inside the `\caption` command. The `\caption[Big Box]` option specifies that the text "Big Box" should be used in the list of figures (where the extra vertical space is not desired).

3.3 Side-by-Side Subfigures

It is often desirable to refer to side-by-side graphics both individually and as a group. The `\subfigure` command (from the subfigure package) defines the group of side-by-side graphics as a single figure and defines each graphics as a subfigure. For example,

```
\begin{figure}
 \centering
 \subfigure[Small Box with a Long Caption]%
  \label{fig:subfig:a} %% first subfigure
  \includegraphics[width=1.0in]{box.eps}}%
 \hspace{1in}%
 \subfigure[Big Box]{
  \label{fig:subfig:b} %% second subfigure
  \includegraphics[width=1.5in]{box.eps}}
 \caption{Two Subfigures}
 \label{fig:subfig}    %% entire figure
\end{figure}
```

produces Figure 15; label `{fig:subfig:a}` refers to subfigure 15(a), label `{fig:subfig:b}` refers to subfigure 15(b), and label `{fig:subfig}` refers to Figure 15.

3.3.1 Subfigures Inside minipage Environments

Like other side-by-side graphics, subfigures are often put inside `minipage` environments. For example,

```
\begin{figure}
   \centering
   \begin{minipage}[b]{0.5\textwidth}
 \centering
 \subfigure[Small Box with a Long Caption]{
 \label{fig:subfig:mini:a}
 \includegraphics[width=1.0in]{box.eps}}

 \end{minipage}%
   \begin{minipage}[b]{0.5\textwidth}
 \centering
 \subfigure[Big Box]{
 \label{fig:subfig:mini:b}
 \includegraphics[width=1.5in]{box.eps}}

 \end{minipage}
   \caption{Subfigures Inside Minipages}
   \label{fig:subfig:mini}
\end{figure}
```

produces Figure 16 which contains subfigures 16(a) and 16(b).

3.3.2 minipage Environments Inside Subfigures

Since Subfigure 16(a) does not contain anything except the `\includegraphics` command, the caption in subfigure 16(a) is only as wide as the included graphic. If the subfigure instead consists of the entire minipage, the caption is made as wide as the minipage. For example,

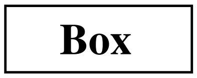


Figure 12: Small Box with a Long Caption

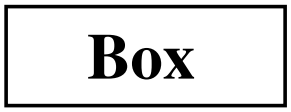


Figure 13: Medium Box

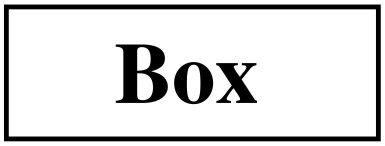


Figure 14: Big Box

(a) Small Box with a Long Caption

(b) Big Box

Figure 15: Two Subfigures

(a) Small Box with a Long Caption

(b) Big Box

Figure 16: Subfigures Inside Minipages

```
\begin{figure}
\subfigure[Small Box with  a Long
   Caption]{\label{fig:mini:subfig:a}
 \begin{minipage}[b]{0.5\textwidth}
   \centering
   \includegraphics[width=1in]{box.eps}
 \end{minipage}}%
\subfigure[Big Box]{
\label{fig:mini:subfig:b}
 \begin{minipage}[b]{0.5\textwidth}
   \centering
   \includegraphics[width=1.3in]{box.eps}
 \end{minipage}}
\caption{Minipages Inside Subfigures}
\label{fig:mini:subfig}
\end{figure}
```

produces Figure 17. Note that the caption of subfigure 17(a) is considerably wider than that of subfigure 16(a).

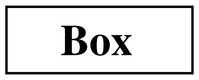
3.3.3 Changing Subfigure Numbering

The subfigure labels have two forms:

1. One which appears under the subfigure as part of the caption, produced by `\@thesubfigure`.
2. One which appears when the `\ref` command is used, produced by concatenating the output of `\p@subfigure` to the output `\thesubfigure`.

These commands use the `subfigure` counter and the `\thefigure` command, making the subfigure label formatting be controlled by the following commands:

- `\thefigure` prints the current figure number.
- The counter `subfigure` counts the subfigures. `\alph{subfigure}` prints the value of the `subfigure` counter in lowercase letters. `\roman{subfigure}` prints the value of the `subfigure` counter in lowercase Roman numerals. (See [1, page 98] or [2, page 446] for a list of counter output commands.)
- `\thesubfigure` is (`\alph{subfigure}`) by default, which produces (a), (b), etc.
- `\@thesubfigure` `\thesubfigure\space` by default, which adds a space between the caption label and the caption.


(a) Small Box with a Long Caption

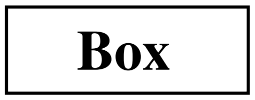

(b) Big Box

Figure 17: Minipages Inside Subfigures

- `\p@subfigure` is `\thefigure` by default

These commands make the default caption labels (a), (b), etc. and the default `\ref` labels 12(a), 12(b), etc. See [10] for controlling the size and font of the subfigure labels.

3.3.4 Subfigure Examples

1. To make the caption labels (i), (ii), etc. and make the `\ref` labels 12i, 12ii, etc., enter the following commands (preferably in the LaTeX file's preamble):

```
\renewcommand{\thesubfigure}
   {\roman{subfigure}}
\makeatletter
\renewcommand{\@thesubfigure}
   {(\thesubfigure)\space}
\renewcommand{\p@subfigure}
   {\thefigure}
\makeatother
```

   `\makeatletter` and `\makeatother` protect the `@` signs in the `\renewcommand` statements.

2. To make the caption labels 12.1:, 12.2:, etc. and make the `\ref` labels 12.1, 12.2, etc., enter the following commands:

```
\renewcommand{\thesubfigure}
  {\thefigure.\arabic{subfigure}}
\makeatletter
\renewcommand{\@thesubfigure}
  {\thesubfigure:\space}
\renewcommand{\p@subfigure}{}
\makeatother
```

3.3.5 Adding Subfigures to List of Figures

The List of Figures generated by `\listoffigures` includes only figures by default, *not* subfigures. To add the subfigures to the List of Figures, type

```
\setcounter{lofdepth}{2}
```

before the `\listoffigures` command.

4 Minipage Placement Option Details

The manner in which `minipage` environments are vertically aligned may be confusing. For example, one might think the commands

```
\begin{figure}
 \centering
 \begin{minipage}[b]{.25\textwidth}
  \centering
  \includegraphics[width=1in]{box.eps}
 \end{minipage}
 \begin{minipage}[b]{.25\textwidth}
  \centering
  \includegraphics[width=1in,angle=-90]{box.eps}
 \end{minipage}
 \caption{\texttt{minipage} with
    \texttt{[b]} option}
\end{figure}
```

which use the minipage `[b]` option would align the bottoms of the graphics. Instead they produce Figure 18.

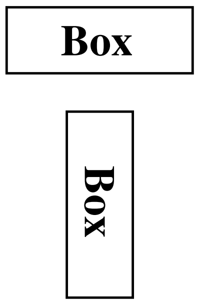


Figure 18: `minipage` with `[b]` option

Similarly, one might think the commands

```
\begin{figure}
 \centering
 \begin{minipage}[t]{.25\textwidth}
  \centering
  \includegraphics[width=1in]{box.eps}
 \end{minipage}
 \begin{minipage}[t]{.25\textwidth}
  \centering
  \includegraphics[width=1in,
          angle=-90]{box.eps}
 \end{minipage}
 \caption{\texttt{minipage} with
    \texttt{[t]} option}
\end{figure}
```

which use the minipage `[t]` options would align the tops of the graphics. Instead they produce a figure which is *exactly* the same as Figure 18.

The `[b]` and `[t]` options produce the same figure because the `minipage` environment's `[b]` option does *not* align the bottoms of the minipages. Rather, it aligns the baselines of the minipages' bottom lines. Similarly, the `[t]` option aligns the baselines of the minipages' top lines. Since the minipages in the above examples only have one line, the `[t]` and `[b]` use the same line for alignment. In this case, the reference point of the minipage is the reference point (original lower-left corner) of the EPS graphic.

4.1 Aligning the Bottoms of Minipages

One method for aligning the bottoms of minipages is to make the bottom of the minipage be the baseline of the minipage. If a line with zero height and zero depth is added inside the minipage after the graphics then the `[b]` option makes the bottom of the minipage be the minipage's baseline. The command `\par\vspace{0pt}` creates such a zero-height, zero-depth line. Since the baseline of this zero-depth line is the bottom of the minipage, the `[b]` option now aligns the bottom of the minipage. For example,

```
\begin{figure}
 \centering
 \begin{minipage}[b]{.25\textwidth}
  \centering
  \includegraphics[width=1in]{box.eps}
  \par\vspace{0pt}
 \end{minipage}
 \begin{minipage}[b]{.25\textwidth}
  \centering
  \includegraphics[width=1in,
      angle=-90]{box.eps}
  \par\vspace{0pt}
 \end{minipage}
 \caption{Minipages with Bottoms Aligned}
\end{figure}
```

produces Figure 19.

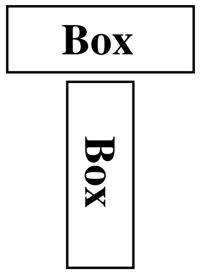


Figure 19: Minipages with Bottoms Aligned

4.2 Aligning the Tops of Minipages

To align the tops of the minipages, one must add a zero-height, zero-depth line to the top of the minipage. Then the `[t]` option makes the top of the minipage be the baseline of the minipage. Preceding `\includegraphics` command by `\vspace{0pt}` inserts a zero-height, zero-depth line above the graphic. Since the baseline of this zero-height line is the top of the minipage, the `[t]` option now aligns the top of the minipage. For example,

```
\begin{figure}
 \centering
 \begin{minipage}[t]{.25\textwidth}
  \vspace{0pt}
  \centering
  \includegraphics[width=1in]{box.eps}
 \end{minipage}
 \begin{minipage}[t]{.25\textwidth}
  \vspace{0pt}
  \centering
  \includegraphics[width=1in,angle=-90]{box.eps}
 \end{minipage}
 \caption{Minipages with Tops Aligned}
\end{figure}
```

produces Figure 20.

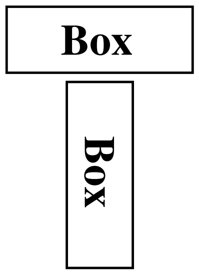


Figure 20: Minipages with Tops Aligned

This aligns the tops of the minipages with the current baseline. If you prefer to align the tops of the minipages with the top of the current line of text, use `\vspace{-\baselineskip}` instead of `\vspace{0pt}`. This topic is discussed in [2, pages 456-457].

5 Boxed Figures

The term *Boxed Figure* usually refers to one of two situations:

- A box surrounds the figure's graphic but not the figure's caption.
- A box surrounds the figure's graphic and its caption.

The basic method for boxing an item is to simply place the item inside an `\fbox` command, which surrounds the object with a rectangular box. The fancybox package provides boxes of different styles.

5.1 Box Around Graphic

Placing `\includegraphics` inside an `\fbox` command produces a box around the included graphic. For example, the commands

```
\begin{figure}
 \centering
 \fbox{\includegraphics
        [totalheight=2in]{file.eps}}
 \caption{Box Around Graphic,
      But Not Around Caption}
 \label{fig:boxed_graphic}
\end{figure}
```

place a box around the included figure, as shown in Figure 21.

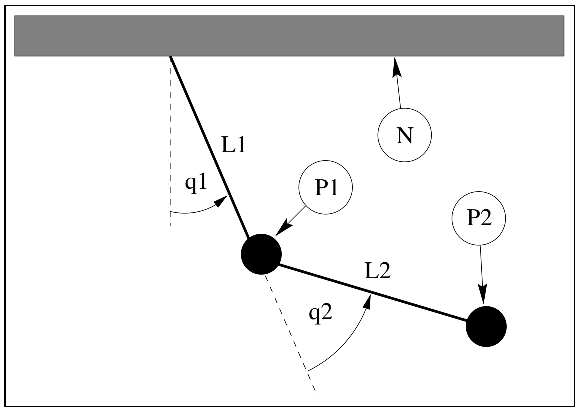


Figure 21: Box Around Graphic, But Not Around Caption

5.2 Box Around Figure and Caption

To include both the figure's graphic and its caption, one may be tempted to move the `\caption` command inside the `\fbox` command. However, this does not work because `\caption` can only be used in paragraph mode, while the contents of an `\fbox` command are processed in LR mode. (LaTeX uses three modes: LR mode, paragraph mode, and math mode. See [1, pages 36,103-5] for an explanation.)

Since the contents of minipage environments and `\parbox` commands are processed in paragraph mode, the `\caption` command can be included in the `\fbox` by enclosing the `\fbox` contents inside a minipage environment or a `\parbox` command. Since both minipages and parboxes require a width specification, there is no direct way to make the `\fbox` exactly as wide as the graphic and caption.

For example, the commands

```
\begin{figure}
 \centering
 \fbox{
  \begin{minipage}{3in}
  \centering
  \includegraphics
       [totalheight=2in]{pend.eps}
  \caption{Box Around Figure
         Graphic and Caption}
  \label{fig:boxed_figure}
 \end{minipage}
 }
\end{figure}
```

place a box around the figure's graphic and caption, as shown in Figure 22.

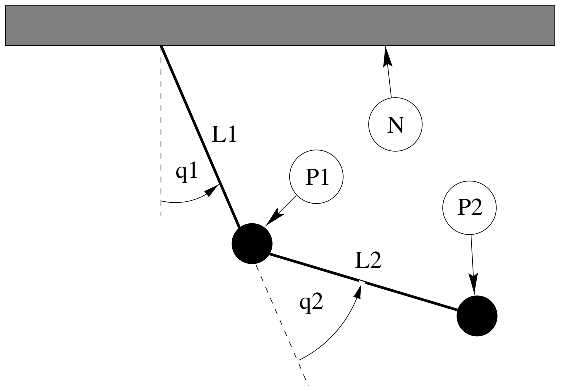


Figure 22: Box Around Figure Graphic and Caption

The determination of a proper minipage width is usually a trial-and-error process. If the caption is wider than the graphic, the minipage can be made as wide as the caption by estimating the caption width with a `\settowidth` command:

```
\begin{figure}
 \centering
 \newlength{\mylength}
 \settowidth{\mylength}
   {Figure XX: Box Around
     Figure Graphic and Caption}
 \fbox{
 \begin{minipage}{\mylength}
  \centering
  \includegraphics[totalheight=2in]{pend.eps}
  \caption{Box Around Figure Graphic
     and Caption}
   \label{fig:boxed_figure_length}
 \end{minipage}
 }
\end{figure}
```

5.3 Customizing \fbox Parameters

In Figures 21 and 22, the box is constructed of 0.4 pt thick lines with a 3 pt space between the box and the graphic. These two dimensions can be customized by setting the LaTeX length variables \fboxrule and \fboxsep, respectively, with the \setlength command. For example, the commands

```
\begin{figure}
 \centering
 \setlength{\fboxrule}{3pt}
 \setlength{\fboxsep}{1cm}
 \fbox{\includegraphics
       [totalheight=1.5in]{pend.eps}}
 \caption{Graphic with Customized Box}
 \label{fig:boxed_custom}
\end{figure}
```

place a box with 3 pt thick lines which is separated from the graphic by 1 centimeter, as shown in Figure 23.

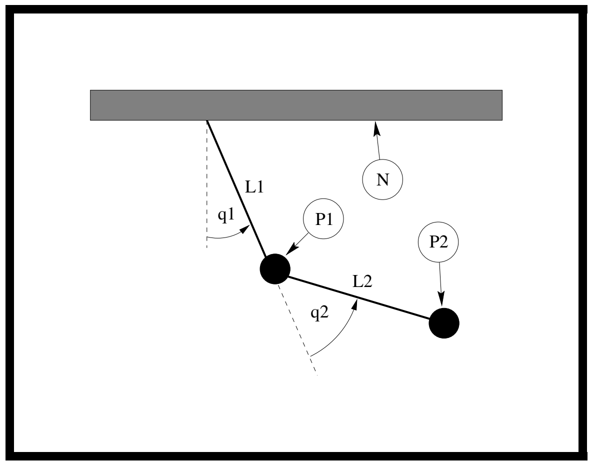


Figure 23: Graphic with Customized Box

5.4 The fancybox Package

In Figures 21, 22, and 23, the \fbox command was used to place standard rectangular boxes around the figures. Alternatively, you can use the fancybox package, which provides the commands \shadowbox, \doublebox, \ovalbox, and \Ovalbox to produce other types of boxes. Details of the commands are given in Table 1.

Like \fbox, the separation between these boxes and their contents is controlled by the LaTeX length \fboxsep. The length \shadowsize is set with the \setlength command, as was done for \fboxrule and \fboxsep in section 5.3. The lines for \ovalbox and \Ovalbox have thicknesses corresponding to the picture environment's \thickline and \thinline, which are *not* lengths and thus cannot be changed with the \setlength command. The values of \thickline and \thinline depend on the size and style of the current font. Typical values are 0.8 pt for \thickline and 0.4 pt for \thinline.

For example, the commands

```
\begin{figure}
  \centering
  \shadowbox{ \begin{minipage}{2.8 in}
 \centering
 \includegraphics[totalheight=1.5in]{pend.eps}
 \caption{Shadowbox Around Entire Figure}
 \label{fig:boxed_fancy}
 \end{minipage} }
\end{figure}
```

place a shadow box around the figure's graphic and caption, as shown in Figure 24.

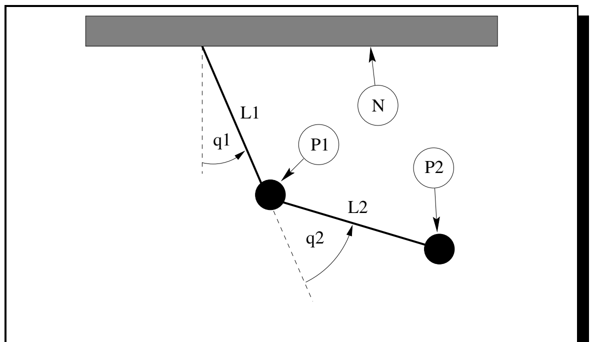


Figure 24: Shadowbox Around Entire Figure

6 Customizing Captions

6.1 Captions Next to Figures

The \caption command places the caption under the figure or table. Minipage environments can be used to trick the caption command into placing the caption next to the figure. For example, the commands

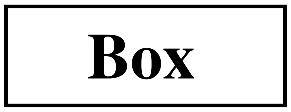
```
\begin{figure}
 \centering
 \begin{minipage}[c]{1.5in}
 \centering
 \caption{Caption on the Side}
 \label{fig:side:caption}
 \end{minipage}
 \hfill
 \begin{minipage}[c]{1.5in}
  \centering
  \includegraphics[width=\textwidth]{box.eps}
 \end{minipage}
\end{figure}
```

produces Figure 25. Likewise, the caption can be placed to the right of the figure by changing the order of the minipages.

Table 1: Fancybox commands

Command	**Parameters**
`\shadowbox{Example}` Example	The frame thickness is `\fboxrule`. The shadow thickness is `\shadowsize` (which defaults to 4 pt).
`\doublebox{Example}` Example	The inner frame thickness is .75`\fboxrule` and the outer frame thickness is 1.5`\fboxrule`. The spacing between the frames is 1.5`\fboxrule` + 0.5pt.
`\ovalbox{Example}` Example	The frame thickness is `\thinlines`. Entering `\cornersize{x}` makes the diameter of the corners `x` times the minimum of the width and the height. The default is `\cornersize{0.5}`. The corner diameter can be set directly by `\cornersize*` command. For example, `\cornersize*{1cm}` makes the corner diameters 1 cm.
`\Ovalbox{Example}` Example	`Ovalbox` is exactly the same as `ovalbox` except that the line thickness is controlled by `\thicklines`.

Figure 25: Caption on the Side



Because the figure environment defined by the float package places the caption *below* the body, Figure 25 cannot be produced with the float package's figure environment. Other parts of the float package can be used as long as `\restylefloat{figure}` is not issued.

6.2 Controlling Caption Width

Since placing the `\caption` command inside a minipage environment makes the caption as wide as the minipage, this can be used to control the caption width. For example, the commands

```
\begin{figure}
  \centering
  \includegraphics[width=2in]{box.eps}
  \caption{Graphic with a Very, Very, Very,
    Very, Very, Very Long Caption}
\end{figure}
```

produce the graphic in Figure 26.

Note that the caption in Figure 26 is as wide as the page text. The width of the caption can be

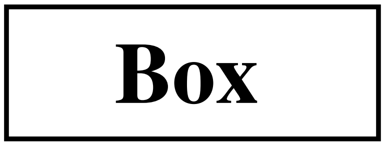


Figure 26: Graphic with a Very, Very, Very, Very, Very, Very Long Caption

limited by placing it inside a minipage environment. For example, the commands

```
\begin{figure}
 \centering
 \begin{minipage}{2in}
  \centering
  \includegraphics[width=1.5in]{box.eps}
  \caption{Graphic with a Very, Very, Very,
     Very, Very, Very Long Caption}
 \end{minipage}
\end{figure}
```

produce the graphic in Figure 27. The minipage limits the width of the caption in Figure 27 to 2 inches.

A more general approach to controlling caption width is provided by the caption package, described in section 6.3.5.

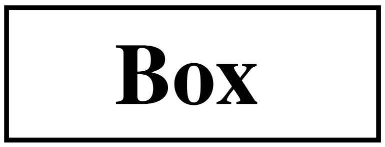


Figure 27: Graphic with a Very, Very, Very, Very, Very, Very Long Caption

6.3 The caption Package

Since the format of LaTeX figure and table captions (especially for multi-line captions) may not be exactly what users desire, the caption package was written by Harald Axel Sommerfeldt to add flexibility to the caption formatting. Since the original caption package had some bad side-effects (particularly the requirement that it be loaded *after* other packages) it was totally re-written and renamed caption2. Although the caption2 package is technically still a beta version, it is quite stable and performs well.

The caption2 package can be used with many types of floats as it directly supports the float, longtable, and subfigure packages, and also works with the floatfig, rotating, supertabular, and wrapfig packages.

Reference [13] describes the commands for the original caption package, while the caption2 reference [14] currently includes only minimal documentation. The `test2.tex` test file demonstrates many of the caption2 capabilities. The options are described in Table 2.

6.3.1 Caption Styles

The caption2 package defines the following caption styles:

normal Full lines are justified (aligned with both left and right margins) with the last line being left-justified.

center All lines of the caption are centered.

flushleft All lines of the caption are left-justified, leaving the right side ragged.

flushright All lines are right-justified, leaving the left side ragged.

centerlast All the lines are justified with the last line being centered.

indent Same as "normal" style except that the second and subsequent lines are indented by the length `\captionindent`. `\captionindent` is zero by default, so a command such as

```
\setlength{\captionindent}{1cm}
```

must be used to set the indentation.

hang Same as "normal" style except that the second and subsequent lines are indented by the width of the caption label (e.g., "Figure 12:").

Usually these styles are specified as `\usepackage` options such as

```
\usepackage[centerlast]{caption2}
```

which makes all the captions in the document have `centerlast` style. Examples of the caption styles are shown in Figures 28–34.

6.3.2 Changing the Caption Style

The `\captionstyle` command changes the caption style. Placing the `\captionstyle` command inside an environment changes only those captions in that environment. For example, the commands

```
\begin{figure}
\captionstyle{centerlast}
\centering
\includegraphics[width=3in]{box.eps}
\caption{Centerlast Caption Style.
   Centerlast Caption Style.}
\end{figure}
```

give only the current figure a `centerlast` style because `\captionstyle` is inside the figure environment. The commands

```
\captionstyle{centerlast}
\begin{figure}
 \centering
 \includegraphics[width=3in]{box.eps}
 \caption{Centerlast Caption Style.
    Centerlast Caption Style.}
\end{figure}
```

give subsequent figures a `centerlast` style because `\captionstyle` is outside the figure environment.

6.3.3 One-Line Captions

If the caption is only one line, all of the above styles center the caption. To force the styles to be enforced even for one-line captions, one must include the `nooneline` option:

```
   \usepackage[nooneline,flushleft]{caption2}
```

This formats *all* captions (including one-line captions) with the `flushleft` style. If you want to change the `nooneline` option inside the document, `\onelinecaptionstrue` centers one-line captions, and `\onelinecaptionsfalse` formats them as normal. For example, the commands

```
\begin{figure}
 \captionstyle{flushleft}
 \onelinecaptionstrue
 \centering
 \begin{minipage}[c]{2.5in}
```

Table 2: `caption2` options

Caption Style	`normal`, `center`, `flushleft`, `flushright`, `centerlast`, `hang`, `indent`	Selects the caption style (see section 6.3.1).
Caption Fontsize	`scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`	Select the fontsize for the caption label (e.g., "Figure 12:") and the caption text.
Caption Label Font Shape	`up`, `it`, `sl`, `sc`	Makes the caption label (e.g., "Figure 12:") have upright, italic, slanted, or small caps shape, respectively. Does not affect caption text.
Caption Label Font Series	`md`, `bf`	Makes the caption label (e.g., "Figure 12:") have a medium or boldface series font, respectively. Does not affect caption text.
Caption Label Font Family	`rm`, `sf`, `tt`	Makes the caption label (e.g., "Figure 12:") have roman, sans serif, or typewriter font, respectively. Does not affect caption text.
One-Line Caption Formatting	`oneline`, `nooneline`	Controls the formatting for one-line captions (see section 6.3.3)

```
   \includegraphics[width=\textwidth]{box.eps}
   \caption{First Caption}
 \end{minipage}
\end{figure}
```

center one-line captions as shown in Figure 35.
The commands

```
\begin{figure}
 \captionstyle{flushleft}
 \onelinecaptionsfalse
 \centering
 \begin{minipage}[c]{2.5in}
   \includegraphics[width=\textwidth]{box.eps}
   \caption{Second Caption}
 \end{minipage}
\end{figure}
```

format one-line captions as shown in Figure 36.

6.3.4 Linebreaks in Captions

When the caption fits in one line, it is processed in an `hbox`, which ignores any `\\` or `\par`. Thus one cannot generally specify linebreaks in captions. However, the caption2 package provides the `\onelinecaptionsfalse` command (or `nooneline` option) to turn off this behavior. For example, the commands

```
\begin{figure}
 \centering
 \includegraphics[width=3in]{box.eps}
 \captionstyle{center}
 \onelinecaptionsfalse
 \caption{First Line of Caption
   \protect\\ Second Line of Caption}
 \label{fig:caption:linebreak}
\end{figure}
```

produce the caption in Figure 37. Since `\\` is fragile, it must be preceded by `\protect`.

6.3.5 Caption Widths

Section 6.2 demonstrated that a `\caption` command appearing in outer paragraph mode can become as wide as the page text as shown in Figure 26. Placing a `\caption` command in a minipage limits the width of the caption to the width of the minipage as shown in Figure 27. The caption2 package provides functions which directly specify the captions' width/margins.

- `\setcaptionwidth{`*`width`*`}` sets the width of the caption to *width*, where *width* can be in any valid TEX units.
- `\setcaptionmargin{mar}` sets the margins to *mar*, making the caption width be the standard width minus 2 times *mar*.

  If *mar* is negative, the caption is made wider than the standard width, which is useful in subfigures and minipage environments.

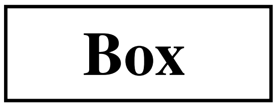


Figure 28: Normal Caption Style. Normal Caption Style.

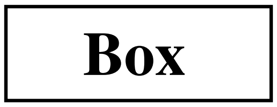


Figure 29: Center Caption Style. Center Caption Style.

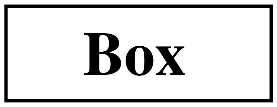


Figure 30: Centerlast Caption Style. Centerlast Caption Style.

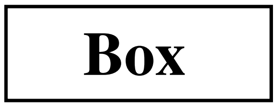


Figure 31: Flushleft Caption Style. Flushleft Caption Style.

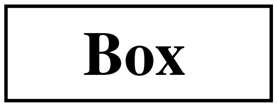


Figure 32: Flushright Caption Style. Flushright Caption Style.

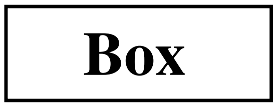


Figure 33: Indent Caption Style. Indent Caption Style.

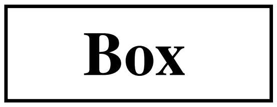


Figure 34: Hang Caption Style. Hang Caption Style.

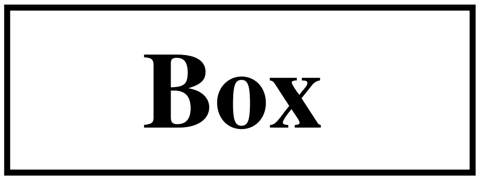


Figure 35: First Caption

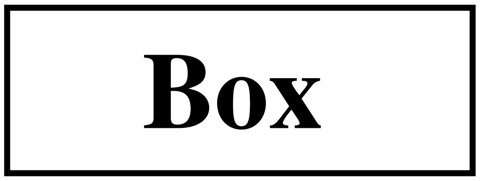


Figure 36: Second Caption

For example, the commands

```
\begin{figure}
 \setcaptionwidth{3in}
 \centering
 \includegraphics[width=2in]{box.eps}
 \caption{Figure Caption Limited to Three Inches}
\end{figure}
```

make the caption 3 inches wide, as shown in Figure 38.

While in the previous example we directly set the width of the caption, alternatively the width can be indirectly set by specifying the caption's margin. For example, the commands

```
\begin{figure}
 \captionstyle{normal}
 \setcaptionmargin{1in}
```

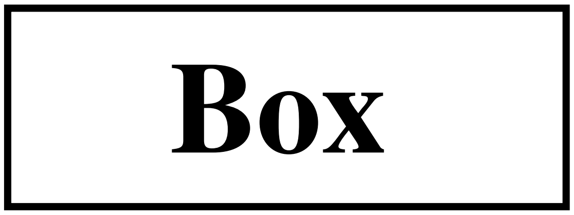


Figure 37: First Line of Caption
Second Line of Caption

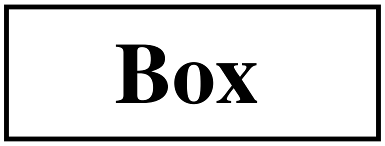


Figure 38: Figure Caption Limited to Three Inches

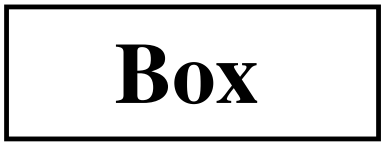


Figure 39: Figure Caption With One-Inch Margins on Each Side

```
 \centering
 \includegraphics[width=2in]{box.eps}
 \caption{Figure Caption With
   One-Inch Margins on Each Side}
\end{figure}
```

indent both sides of the caption one inch from the page margins, as shown in Figure 39.

6.3.6 Caption Font and Delimiter

While the `scriptsize`, `Large` and other options for `\usepackage{caption2}` change the size of both the caption label (e.g., “Figure 12:”) and the caption text, the `up`, `it`, `sl`, `sc`, `md`, `bf`, `rm`, `sf`, and `tt` options affect only the caption label.

Users can achieve more flexibility by redefining the `\captionfont` and `\captionlabelfont` commands. The caption is created by the following commands

```
 {\captionfont%
 {\captionlabelfont \captionlabel
    \captionlabeldelim}%
 \captiontext}
```

where the `\captionlabel` command produces “Figure 12”, the `\captionlabeldelim` command produces “:”, and the `\captiontext` command produces the caption text. Thus `\captionfont` affects both the caption label and caption text, while `\captionlabelfont` affects only the caption label.

LaTeX fonts are described by encoding, size and three type style components: shape, series, and family ([1, pages 37,115], [2, pages 170-71]). These characteristics can be specified in the `\captionfont` and `\captionlabelfont` commands. For example, the commands

```
\begin{figure}
 \renewcommand{\captionfont}
    {\Large \bfseries \sffamily}
 \renewcommand{\captionlabelfont}{}
 \centering
 \includegraphics[width=2in]{box.eps}
 \caption{Test Caption}
\end{figure}
```

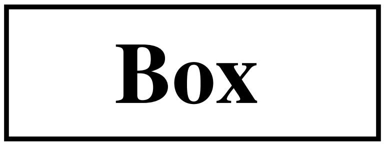


Figure 40: Test Caption

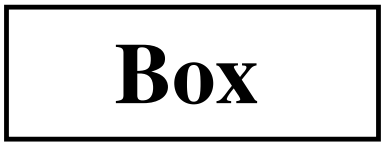


Figure 41: **Test Caption**

produce Figure 40. Here the `\captionlabelfont` command does nothing. This means that it does not overwrite any font characteristics and all the `\captionfont` settings are carried over to the caption label. Since no shape declaration was specified, the entire caption has the default upright shape. The commands

```
\begin{figure}
 \captionstyle{normal}
 \renewcommand{\captionfont}
   {\Large \bfseries \sffamily}
 \renewcommand{\captionlabelfont}{\small}
 \centering
 \includegraphics[width=2in]{box.eps}
 \caption{Test Caption}
\end{figure}
```

produce Figure 41. In this example, the `\small` in `\captionlabelfont` overwrites the `\Large` from `\captionfont`. However, since there are no series or family changes in `\captionlabelfont`, the `\bfseries` and `\sffamily` declarations carry over to the caption label.

The default colon delimiter can be changed by redefining the `\captionlabeldelim` function. For example, the commands

```
\begin{figure}
 \captionstyle{normal}
 \renewcommand{\captionlabeldelim}{.\quad}
 \centering
 \includegraphics[width=2in]{box.eps}
 \caption{Caption with New Delimiter}
\end{figure}
```

change the delimiter in Figure 42 from the default colon to a period followed by a quad space.

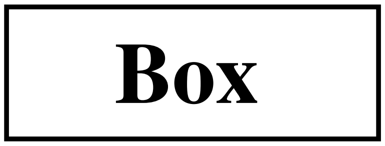


Figure 42. Caption with New Delimiter

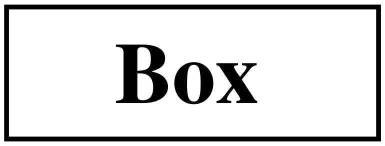


Figure 43
Customized Caption Style

6.3.7 Custom Caption Styles

The caption2 package also allows users to create their own caption styles. For example, the following commands

```
\newcaptionstyle{mystyle}{%
\usecaptionmargin\captionfont%
{{\centering\bfseries
  \captionlabelfont\captionlabel\par}%
\centering\captiontext\par}}

\begin{figure}
\captionstyle{mystyle}
\centering
\includegraphics[width=2in]{box.eps}
\caption{Customized Caption Style}
\end{figure}
```

makes the caption label boldface and places it on a separate line from the caption text, as shown in Figure 43.
See the caption2 test file [15] for more user-defined caption style examples.

7 The PSfrag System

While there are many drawing and analysis packages which produce EPS files, most of them do not support symbols and equations as well as LaTeX. The PSfrag system allows LaTeX users to replace text strings in EPS files with LaTeX text or equations. Currently available for both DOS and Unix, the PSfrag system consists of the LaTeX style file `psfrag.sty` and the `perl` script `ps2frag` and is well-documented by [11].

PSfrag currently does not support compressed or non-EPS graphics. This means that if PSfrag is used for even one graphic in a document, all of the document's graphics must be non-compressed EPS files.

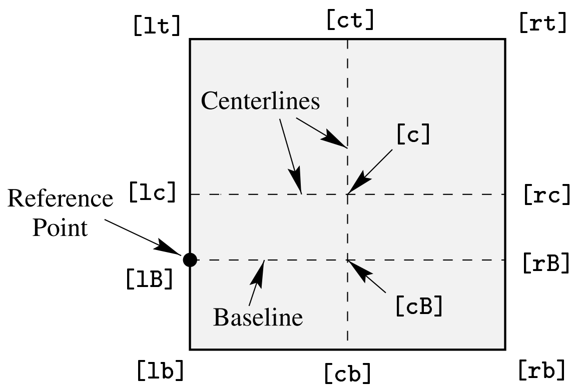


Figure 44: Available Origin Points

The procedure for using `PSfrag` is:

1. Create an EPS file.
2. At the operating system prompt, type:
   `ps2frag file.eps`
   which scans `file.eps` for text strings and then records these locations in the EPS file. Since this added information is in the form of header comments in the EPS file, it does not change the appearance of the EPS output.
3. In the LaTeX document, use the following commands:
   (a) Include `\usepackage{psfrag}` in the preamble.
   (b) Use the `\psfrag` command to specify the EPS text and the LaTeX string to replace it. This makes the specified substitution occur in any subsequent `\includegraphics` command issued in the same environment.
   (c) Use the `\includegraphics` command as usual.

The LaTeX `\psfrag` command has the following syntax

```
\psfrag{PStext}[posn][PSposn][scale][rot]{text}
```

with its arguments described in Table 3.

The `posn` and `PSposn` options are one of the 12 points shown in Figure 44 on page 301, except that the `c` specifier is not available (e.g., to align the left-center, use `[l]` instead of `[lc]`; to align centers, use `[]` instead of `[cc]`). See [11] for examples of various combinations of placement points.

7.1 PSfrag Example

The commands

```
\includegraphics{pend.eps}
```

include the graphic without any PSfrag replacement, producing Figure 45. The commands

Table 3: `\psfrag` Options

`PStext`	Text in EPS file to be replaced. PSfrag is sensitive about what type of text it replaces. For example, if the EPS file contains the text `Error (%)`, the percent sign confuses LaTeX and PSfrag *cannot* be used on the file, regardless of whether PSfrag replaces `Error (%)`. Instead, regenerate the EPS file using text such as Error (percent) which does not contain any of the LaTeX special characters.
`posn`	*(Optional, Defaults to `[Bl]`.)* Position of placement point relative to new LaTeX text. `[]` indicates center.
`PSposn`	*(Optional, Defaults to `[Bl]`.)* Position of placement point relative to existing EPS text. `[]` indicates center.
`scale`	*(Optional, defaults to 1.)* Scaling factor for the text. For best results, avoid using the scaling factor and instead use LaTeX type-size commands such as `\small` and `\large`.
`rot`	*(Optional, defaults to zero.)* When this rotation angle is zero, the new text is inserted at the same angle as the existing EPS text. When an angle is specified here, it is the angle of rotation of the new text relative to the existing text. The angle is in degrees with a counterclockwise rotation being positive. This option is useful when dealing with EPS files generated by applications which only allow horizontal text. This option effectively adds rotated-text capabilities to those applications.
`text`	The LaTeX text to insert into the EPS graphic. Like regular LaTeX text, math formulas must be enclosed by dollar signs (e.g., `$\frac{1}{2}$` or `$x^2$`) and special symbols can be used (e.g., `\%` produces %).

```
\psfrag{q1}{$\theta_1$}
\psfrag{q2}{$\theta_2$}
\psfrag{L1}{$L_1$}
\psfrag{L2}{$L_2$}
\psfrag{P1}[][]{$P_1$}
\psfrag{P2}[][]{\Large $P_2$}
\includegraphics{pend.eps}
```

include the graphic with PSfrag replacement, producing Figure 46. The first four `\psfrag` commands position the new LaTeX text such that its left baseline point corresponds to the left baseline point of the EPS text. The last two `\psfrag` commands position the new LaTeX text such that its center corresponds to the center of the EPS text.

Note that one need not replace all the EPS text with LaTeX text. For example, the `N` tag is left unchanged in Figure 46. Also note that `\psfrag` matches *entire* text strings. Thus the command

```
\psfrag{pi}{$\pi$}
```

replaces the string `pi` with π, but does not affect the strings `pi/2` or `2pi`. Separate `\psfrag` commands must entered for these strings.

7.2 LaTeX Text in EPS File

In the previous section, the `\psfrag` command specified the LaTeX text in the LaTeX file. While this is the most popular method, PSfrag's `\tex` command includes LaTeX text directly in EPS files. The `\tex` command has the following syntax

```
\tex[posn][PSposn][scale][rot]{text}
```

which is the same as `\psfrag` command, except that there is no `{PStext}` argument. Unlike the `\psfrag` command, the `\tex` command is placed in the EPS file.

For example, if an EPS file contains the text `\tex{$x^2$}` PSfrag automatically replaces it with x^2. The left-baseline point of x^2 is aligned with the left-baseline point of `\tex{$x^2$}`. Note that PSfrag does the replacement automatically; apart from the `\usepackage{psfrag}` command, it does not require any commands in the LaTeX file. Placement, scaling, and rotation options can be specified as with the `\psfrag` command. If an EPS file contains the text `\tex[][]{$x^2$}` PSfrag replaces it with a centered x^2. The `\tex` command must be *entire* text string. For example, the text

```
Transfer Function \tex{$\frac{s+a}{s+b}$}
```

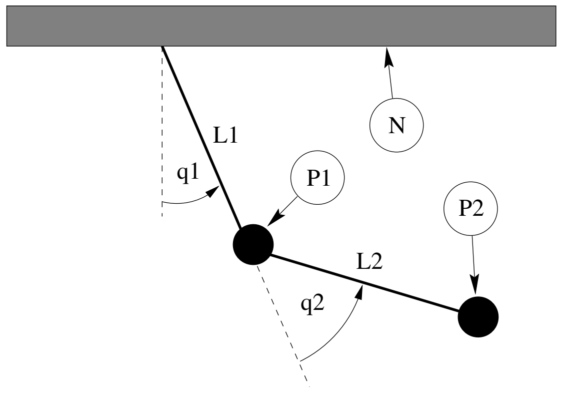


Figure 45: Without PSfrag Replacement

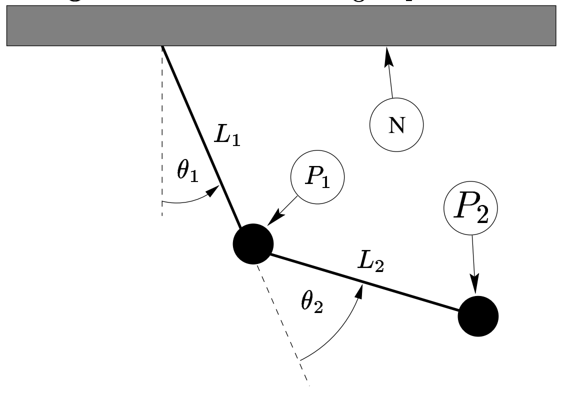


Figure 46: With PSfrag Replacement

produces an error. Instead use

```
\tex{Transfer Function $\frac{s+a}{s+b}$}
```

The advantage of the `\tex` command is that the LaTeX file doesn't need to be edited when an EPS file is modified. The `\tex` command has two disadvantages. First, the EPS file cannot be used for non-LaTeX purposes, while the EPS graphic in Figure 45 could be used without replacement. Second, if `\tex` command contains complicated formulas, the text can extend beyond the edge of the graphics, enlarging the EPS BoundingBox. This oversized BoundingBox causes incorrect graphic placement in LaTeX.

7.3 Text Scaling in PSfrag

A subtlety of the `\includegraphics` command (see [6, page 3]) comes into play with PSfrag. When scaling options are specified *before* rotation

```
\includegraphics[width=3in,angle=30]{file.eps}
```

the scaling is implicitly handled by the graphics inclusion function. However, when scaling options are specified *after* rotation

```
\includegraphics[angle=30,width=3in]{file.eps}
```

the graphic is first included at its natural size, then rotated, and then scaled. Since PSfrag replaces the new text during the graphics inclusion, the second command scales the new PSfrag text while the first command does *not*. When the included size of the EPS graphic greatly differs from its natural size, the two commands produce very different results. See [11, pages 10-11] for information.

8 Graphics in Page Header or Footer

The fancyhdr package (an improved version of the old fancyheadings package) makes it easy to customize a document's page headers and footers. It is often desired to place a logo or other EPS graphics in the header or footer, which results in the same EPS file being included multiple times.

8.1 Including An EPS File Multiple Times

There are three common methods for including the same EPS graphics many times

1. Use `\includegraphics{file}` in the places where you want the graphic. This has two problems
   (a) LaTeX must find and read the file every time the `\includegraphics` command is used.
   (b) The repeated graphics commands may result in a very large final PostScript file.
2. Save the graphics in a LaTeX box, using the box wherever you want the graphic. This saves LaTeX time since it must only find and read the file once. However, it does not reduce the size of the final PostScript file.

   At the beginning of the file, include the following commands

   ```
   \newsavebox\mygraphic
   \sbox\mygraphic{\includegraphics{file.eps}}
   ```

   Then use the command `\usebox{\mygraphic}` wherever you want the graphic.
3. Define a PostScript command which draws the graphics, and then issue the PostScript command wherever you want the graphic. Since the final PostScript file includes the graphics commands only once, the final PostScript file is much smaller.

   Since the graphics commands are stored in printer memory while the final PostScript file is being printed, this method may cause the printer to run out of memory and not print the document.

8.2 Defining a PostScript Command

To convert the EPS graphics into a PostScript Command, the EPS file must be broken into two files, one which defines the PostScript dictionary and the graphics commands, and another which includes the header information and uses the previously-defined PostScript command. For example, an EPS file created by xfig has the form

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep  3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog

$F2psBegin
...
$F2psEnd
```

Where ... indicates unlisted commands. The EPS file generally contains three parts

1. The header commands which begin with %
2. The Prolog section which starts with

   `/$F2psDict 200 dict def`

   and ends with `%%EndProlog`. The Prolog defines the commands in the PostScript dictionary used by the EPS file. In this example, the dictionary is named `$F2psDict` although other names can be used.
3. The last part contains the commands used to draw the graphics.

Suppose the above EPS file is named `file.eps`. Now create the files `file.h` and `file.ps` where `file.h` contains

```
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog

/MyFigure {
$F2psBegin
...
$F2psEnd
} def
```

and `file.ps` contains

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep  3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
$F2psDict begin MyFigure end
```

`file.h` defines the dictionary and defines the PostScript command `/MyFigure`, while `file.ps` contains the header information and uses the PostScript command defined in `file.h`. In particular, it is important that the `file.ps` header includes the `%!PS...` line and the BoundingBox line. The graphics can then be used in the LaTeX document as

```
   \documentclass{article}
   \usepackage{graphicx}
   ...
   \special{header=file.h}
   ...
   \begin{document}
...
\includegraphics[width=2in]{file.ps}
...
\includegraphics[totalheight=1in]{file.ps}
...
   \end{document}
```

Note that the original file `file.eps` is not used. Since the graphics commands in `file.h` are only included once, the final PostScript file remains small. However, this still requires LaTeX to find and read `file.ps` whenever the graphics are used. The following commands produce a small final PostScript file while reading `file.ps` only once.

```
\documentclass{article}
\usepackage{graphicx}
...
\special{header=file.h}
\newsavebox\mygraphic
\sbox\mygraphic{
   \includegraphics[width=2in]{file.ps}}
\begin{document}
...
\usebox{\mygraphic}
...
\resizebox*{1in}{!}{\usebox{\mygraphic}}
...
\end{document}
```

Like the previous example, these commands produce a 2-inch wide graphic and another graphic whose totalheight is 1 inch.

8.3 The fancyhdr Package

An easy method of including graphics in the heading is to use the fancyhdr package, which is documented by [8]. The header consists of three parts: its left field, its center field, and its right field. The

\fancyhead command specifies the contents of the header fields, with the L,C,R options specifying the field(s) which the command should modify. For example

```
\pagestyle{fancy}
\fancyhead[C]{My Paper}
```

causes the center header field to be "My Paper", while

```
\pagestyle{fancy}
\fancyhead[L,R]{\textbf{Confidential}}
```

causes the text of left and right header fields to be "**Confidential**". If no L,C,R option is specified, it applies to all three header fields, so \fancyhead{} is used to clear all the header fields. Similarly, \fancyfoot{} specifies the left, center, and right footer fields.

Note that the \fancyhead commands only apply to pages whose style is "fancy". Even though \pagestyle{fancy} causes the document to have a fancy page style, some pages (title pages, table of contents pages, the first page of chapters, etc.) are still given a plain pagestyle by default.

8.3.1 Graphics in Page Header/Footer

The commands in the fancyhdr package can insert graphics in the headers and footers. For example, after splitting the EPS file file.eps into the two file file.h and file.ps as described in section 8.2, the commands

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}
%% must be large enough for graphic
\renewcommand{\headheight}{0.6in}
\renewcommand{\textheight}{7.5in}

% Define PostScript graphics command
\special{header=file.h}

% Save graphics in LaTeX box
\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics
  [totalheight=0.5in]{file.ps}}

\pagestyle{fancy}
\fancyhead{}   % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{}   % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}

\begin{document}
...
\end{document}
```

places the graphics at the top left of each "fancy" page with a 0.5 pt horizontal line drawn under the header. Additionally, the page number is placed at the bottom center of each page, with no horizontal line drawn above the footer.

8.3.2 Odd/Even Headings

When the [twoside] documentclass option is used, one may want to individually specify the odd and even page headers/footers. The E,O \fancyhead options specify the even and odd page headers, respectively. If the E,O options are not specified, the command applies to both even and odd pages. Likewise the E,O \fancyfoot options specify the even and odd page footers. For example,

```
\pagestyle{fancy}
\fancyhead[LE]{My Paper}
\fancyhead[RO]{My Name}
\fancyfoot[C]{\thepage}
```

places "My Paper" in the upper left of even fancy pages, "My Name" in the upper right of odd fancy pages, and the page number in the bottom center of all fancy pages. Replacing the

```
\fancyhead[L]{\usebox{\mygraphic}}
```

command in the above example with

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

places the graphic at the top outside (the left side of even pages, right side of odd pages) of all fancy pages.

8.3.3 Modifying Plain Pages

Although the above commands do not affect pages with plain pagestyles, the \fancypagestyle command can be used to modify the plain pagestyle. For example

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}

%% must be large enough for graphic
\renewcommand{\headheight}{0.6in}
\renewcommand{\textheight}{7.5in}

% Define PostScript graphics command
\special{header=file.h}

% Save graphics in LaTeX box
\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics
    [totalheight=0.5in]{file.ps}}

\pagestyle{fancy}
\fancyhead{}   % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{}   % clear all footer fields
\fancyfoot[C]{\thepage}
```

```
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}

\fancypagestyle{plain}{%
  \fancyhead{}   % clear all header fields
  \fancyhead[L]{\usebox{\mygraphic}}
  \fancyfoot{}   % clear all footer fields
  \fancyfoot[C]{\thepage}
  \renewcommand{\headrulewidth}{0.5pt}
  \renewcommand{\footrulewidth}{0pt}}

\begin{document}
...
\end{document}
```

place the graphic at the upper left of every page (both plain and fancy). Likewise, when the `twoside` documentclass option is used, replacing both of the

```
\fancyhead[L]{\usebox{\mygraphic}}
```

commands with

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

places the graphic at the top outside of every page (both plain and fancy).

Acknowledgements

I would like to thank the contributors to the newsgroup `comp.text.tex`, whose posts and replies provided me with the information for this document. In particular, David Carlisle provided a great deal of assistance. I would also like to acknowledge Jim Hafner for providing the procedure in section 8.2. Finally, I would like to thank the readers of previous versions who provided me with feedback.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1.

[2] Michel Goossens, Frank Mittelbach and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8.

[3] Helmut Kopka and Patrick Daly. *A Guide to LaTeX 2ε.* Addison-Wesley, Reading, Massachusetts, 1995, ISBN 0-201-4277-X.

[4] D. P. Carlisle. *Packages in the 'graphics' bundle.* Available from CTAN as `grfguide.tex` or `grfguide.ps`.

[5] D. P. Carlisle and S. P. Q. Rahtz. *The graphics package.* Available from CTAN as `graphics.dtx`.

[6] D. P. Carlisle and S. P. Q. Rahtz. *The graphicx package.* Available from CTAN as `graphicx.dtx`.

[7] Trevor Darrell. *Psfig/TeX 1.10 Users Guide.* Available from CTAN as `psfig-doc.tex`.

[8] Piet van Oostrum, *Page layout in LaTeX*, Available from CTAN as `fancyhdr.tex`

[9] Sebastian Rahtz and Leonor Barroca, *The `rotating` package*, Available from CTAN as `rotating.dtx`

[10] Steven Douglas Cochran. *The `subfigure` package.* Available from CTAN as `subfigure.dtx`.

[11] Craig Barratt and Michael C. Grant. *The PSfrag system.* Available from CTAN as `pfgguide.tex`.

[12] Timothy Van Zandt. *Documentation for fancybox.sty* Available from CTAN as `fancybox.doc`

[13] Harald Axel Sommerfeldt. *The caption package* Available from CTAN as `caption.dtx`

[14] Harald Axel Sommerfeldt. *The caption package.* Available from CTAN as `caption2.dtx`

[15] Harald Axel Sommerfeldt. *Test of the caption package.* Available from CTAN as `test2.tex`

◇ Keith Reckdahl
Stanford University
Box 9030
Palo Alto, CA 94309
USA
`reckdahl@leland.stanford.edu`

Macros

Fast and secure multiple-option tests

Jordi Saludes

Introduction

Some of us are teachers and have to evaluate the performance of students many times a year. In such cases multiple-option tests are a very common solution and with the introduction of optical mark readers automation of the grading process has become possible. However in case of space or time constraints, such devices give no real help: In a crowded examination room we have to produce several versions of the exam to prevent peeking.[1] So we have to give the reader service a solution sheet for each version. On the other hand when the reader serves a big community the delay in getting the grades is often too large (at least to give feedback to the students).

The convenience of using TEX to typeset exams in different versions by scrambling questions and answers was first addressed in [D]. I tried to go a step further to help those of us that can not rely on special hardware. I propose a procedure to easily make many versions of multiple-option tests, that relieves the teacher of a boring task at the cost of charging the students with a bit more of work.

The teacher prepares the test using a macro package to be described below (see an example in appendices A and B). Running the test file produces tests in many versions.

In these tests, answers are labeled by (apparently random) integer numbers. To answer a test the student, with the help of a pocket calculator, adds the label numbers of the answers he thinks to be right and writes the totals in the bottom of each page. The student identification and page totals are the only marks allowed in the exam. To avoid errors in the addition, label numbers displayed are in fact multiplied by a small factor like $d = 7$ or $d = 13$ (that we call the *detection factor*), thus the total must be also multiple of d.

Since different versions have different label numbers and moreover answers are scrambled, this is a secure protection against copying. Second, since the answers are coded in only a number, entering exams in a computer for grading can be very fast. (See some suggestions at the end of the paper.)

The knapsack problem. It is an ancient puzzle: Given the total weight s of a knapsack and the weight $w_1, \ldots, w_n$ of individual objects, determine which objects are in the bag. In the general case, this problem is hard when the number of objects is large. Given the difference of computational effort on solving the knapsack problem (going from s to the objects) versus stating it (going from objects to s), this problem can be used as a *trapdoor* function, giving a public key cryptosystem [S], [Ro].

In this paper I slightly modify this cryptosystem to adapt it for doing tests in the classroom. Students give their answers by adding the label numbers of chosen options (encrypting by different public keys). To get back the answers, the teacher (who knows the private key) decrypts the total.

In what follows, *knapsack problem* refers to the following slightly more general problem:

$(\mathcal{B}, s)$: *Let* $\mathcal{B} = \{B_1, \ldots, B_N\}$ *be a collection of mutually disjoint sets* B_i *of positive integers. Given* s *a positive integer determine* $b_i \in B_i \cup \{0\}$ *for* $i = 1, \ldots, N$ *such that* $s = b_1 + \ldots + b_N$.

In other words, take an object out of some of the bags in $\mathcal{B}$ such that the total weight be s.

In general $(\mathcal{B}, s)$ has no solution or, given a solution exists, it is possibly not unique. Finding a solution in general amounts to checking all the different sums, which is not feasible when the total number of elements is large.

When applied to the case of student tests, B_i will be the set of label numbers for the answers of question i. Choosing $b_i \in B_i$ means marking the corresponding answer to question i and taking $b_i = 0$ corresponds to skip this question. This way, determining the marked answers from the total s implies the solving of a knapsack problem.

It is important to use a labeling family $\mathcal{B}$ for which the knapsack problem has a unique solution, for otherwise we would be unable to decide among several test markings with the same sum. It is also important that the teacher can easily solve the problem whereas it must be difficult for anyone else. In the following two sections we consider how to manage that.

Mixed-radix sequences. It is clearly true that when

$$\sum_{i=1}^{n} \max B_i < \min B_{n+1}, \tag{1}$$

the knapsack problem has a unique solution (provided it exists) and moreover, it is really easy to solve the problem by comparing s with the elements of $B_1 \cup \ldots \cup B_N$ arranged in decreasing sequence.

Let us construct a such problem with a given number of elements: Suppose we want B_i to have n_i elements for $i = 1, \ldots, N$. Take a sequence w_i recursively defined as

$$\begin{aligned} w_1 &= 1, \\ w_i &= (1 + n_{i-1}) w_{i-1} \quad \text{for } i = 2, \ldots, N. \end{aligned} \tag{2}$$

and set $B_i = \{ j w_i \mid j = 1, \ldots, n_i \}$ for $i = 1, \ldots, N$. It is easy to show that this collection fulfills condition (1). The sequence (w_i) is called a mixed-radix

[1] Peeking is a very popular sport in Spain.

sequence[2] for (n_i). An algorithm for solving the problem using (w_i) will be given below.

Modular arithmetic. We say that integers m and m' are congruent modulo k ($m \equiv m' \pmod{k}$) if and only if $m - m'$ is a multiple of k. Given m a positive integer there is exactly one $0 \leq r < k$ such that $m \equiv r \pmod{k}$. This number is the remainder of the integer division of m by k and will be denoted $m \bmod k$.

We will consider now other knapsack problems related to equation (2). Take

$$k \geq (1 + n_N)w_N \tag{3}$$

and $1 < v < k$ an integer coprime with k (i.e., $\gcd(v, k) = 1$), and consider the problem for $\tilde{\mathcal{B}} = \{\tilde{B}_1, \ldots, \tilde{B}_N\}$ where $\tilde{B}_i = vB_i \bmod k$. Since the map $x \mapsto vx \bmod k$ is additive, it relates weights and totals of $(\mathcal{B}, s)$ with the corresponding ones of $(\tilde{\mathcal{B}}, \tilde{s})$. Therefore the new problem also has the uniqueness property, albeit condition (1) is not longer true. In fact modular multiplication totally distorts the order relation of (1) making the knapsack problem much more difficult.

We will use just this kind of knapsack systems to label the answers of the students sheets. Since, given a sum, there is exactly one combination of answers adding to this total. But, on the other hand, not having the answer labels a clear order it is not feasible to recover the answers from the sum. Thus we can safely have a stack of answered sheets on our desk with the students nosing around.

The problem is far easier for the teacher, since she knows not only the sum $\tilde{s}$ and v but also the key k. To get the answers she has to:

1. Solve $\tilde{s} \equiv vs \pmod{k}$ for s. This implies computing the *multiplicative inverse* $\bar{v}$ of v modulo k, namely a positive integer such that $v\bar{v} \equiv 1 \pmod{k}$.
2. Find $b_1 \ldots, b_n$ in the original problem for s. Since this problem has a mixed-radix sequence $(w_1, \ldots, w_N)$ we can easily find the solution by iterated integer division.

See below in the grading section for a description of the algorithms used.

[2] These sequences appear, for example, in the old British monetary system and in the measure of time [K]: To convert days, hours, minutes and seconds from/to seconds, we consider the sequence 59, 59, 23 that gives $w_1 = 1$, $w_2 = 60$, $w_3 = 3600$, $w_4 = 86400$.

Preparing a test

To make a test, we write a file `foo.tex` such as the one in appendix B. We begin the file with `\input knst.tex` to load the macros and give commands like `\title` or `\date` to complete the header. Now we issue `\plainversion` to instruct TeX we are in *plain mode* and then `\test{`⟨*k*⟩`}` to begin the exam. We choose the key k in the range of TeX integers, its magnitude depending on the number of questions per page wanted: the more questions the larger the value (see the section on choosing numbers). This number (to be kept private) allows the encryption/decryption of the test. Now we give the questions as follows

```
\qtn ⟨question text⟩
 \anw ⟨1st option text⟩
         \anw ⟨2nd option text⟩

   ⋮
\endqtn
```

We mark each right answer using `\Anw` instead of `\anw`. The package will display a warning if there is no marked option or an error if there are many. Remember that in the student versions, questions (and answers inside questions) will be scrambled, so avoid using expressions such 'In the question above'. However we can write 'None of the above' options, by using `\fix`. This command will keep following options (inside the current question) in place.

Anyway, if we want to keep the question order while scrambling answers we can put a `\fixqtn` right after the `\test` command.

We can force a page break with `\newpage` between `\endqtn` and the next `\qtn`. The file ends with `\endtest`.

Making student versions. Now we delete (or comment out) `\plainversion` in `foo.tex` and provide a version number $1 < v < k$ by assigning v to counter `\verno` before the `\test` command. I suggest to take v large since that way the generated label numbers appear as randomly chosen. As stated before, k and v must be coprimes, otherwise we will get an

```
Invalid stepper/version number
```

error. (See also the section on choosing numbers for suggestions on how to take these.)

We are now in *production mode*. Each time we run TeX on this modified file we obtain a student test (version v). Instead of manually changing `\verno` for each different version of the test, we can use `\stepversion` command as explained below.

The mechanism. In both plain and production mode, the package computes the label numbers of answers and uses vertical boxes to compose questions and answers.

In plain mode (when `\plainversion` is issued), label numbers are not displayed and $v = 1$ is assumed. TEX keeps track of w_i's, info about questions, page breaks, and records all this stuff in file `foo.ans` (as `\opts` and `\pagebot` commands).

Note that a page break may occur by

- Key overflow. As soon as $w_i > k$, the page is terminated in order to keep condition (3) true. The question being processed will be placed in the next page. In the log file `foo.log`, this condition is signaled by a '`*`' just before the shipping of the page.
- Normal page completion. This is the result of undisturbed TEX page builder. This case includes forcing page breaks using `\newpage`.

In both cases TEX writes a `\pagebot` line in `foo.ans`. This command will be used later in production mode to determine the page breaks. Each question generates a TEX `\mark` carrying the question number i and corresponding w_i. This information is used to fill the `\pagebot` and to correct future overestimated w_i's.

In production mode, the boxes containing question and answer text are scrambled using the code in [M]: Depending on the parity of a shift register r questions/answers are appended or prepended to the current list. We can use `\SRset` to set r. The value $r = 0$ means no scrambling at all. In this mode `\newpage` instructions are skipped, since page breaking depends only on previously recorded `\pagebot`s.

The macro package deals with the following files:

- `foo.dvi`. In plain mode it contains the plain version of the test with questions numbered and correct options marked. In production mode it contains a scrambled student test with label numbers.
- `foo.ans` is generated at plain step and read on production. It contains relevant information for breaking pages and grading, such as the key k and the detection factor d. For each question it includes the number of options and the position of the correct answer.
- `foo.aux` is an auxiliary file generated by `\stepversion` containing v and r assignments. If we put `\stepversion{`$\langle t\rangle$`}` just before `\endtest`, where $1 < t < k$ is an integer coprime with k, TEX will step both the scrambler r and the current version number by $v \leftarrow t \cdot v \bmod k$, and then it will write commands to set the new value of v and r in `foo.aux`. This way, without having to change `foo.tex`, each new TEX run will give a new test with different scrambling and version number.

To work like that, command `\test` in production mode obtains the value of v as follows:

1. If `\verno` > 0 use this value as v; this branch is taken when we set `\verno` by hand at the beginning of the test;
2. Otherwise search for a file named `foo.aux` and expand it.
3. If this file is not found, read command from console. This option could be useful in environments supporting pipes (like UNIX).

Grading exams

To grade an exam we have to collect the totals and version number v. A program then computes the multiplicative inverse $\bar{v}$ modulo the key k and uses it to decrypt totals and obtain the options that the student chose.

I give now the algorithms to decode and grade exams from the total and version number in the `awk` language. The reasons are twofold: First, this language is easily readable, and second, the program really works without the hassle of declaring variables, opening files, etc. However, I do not recommend this program for real life usage, because it works with only one page (a total), and because computations are done in floating point. In my machine this means exact integer arithmetic if the magnitude of the numbers involved is less than 2^{31}. Thus we can run into a loss of precision when using large keys and/or version numbers and, what is worse, no notification of that loss will be given.

For more serious usage, I have a C implementation which uses a *Binary Extended Euclid Algorithm* to avoid multiplication of large numbers [C, page 18] and it is not limited to a unique page. I intend to submit it to the CTAN archives.

Decoding. Assume we have a data file, with lines containing version number v, total $\tilde{s}$ and an identifier for each student.

```
38353 11073202 First student
2567   4672433 Second student
```

⋮

Concatenate `foo.ans` with the data file and run `awk` using the following code:

First initialize some variables

```
BEGIN   {wa=1;nqn=0;}
```

Read instructions from the ans file getting k, d and computing w_i using (2). The ans file includes a \Key line that gives k and the detection factor d (columns 2 and 3):

```
/Key/     {key=$2; df=$3; next}
/pagebot/ {next} # ignore pagebot
```

It has also an \opts line for each question, which describes the number of options, the index of the correct answer, and the current value of w_i at the moment it was computed. Note that due to differences between the point where w_i is computed and when the TEX page builder activates, the values of the last column in an \opts line can be over-valued; I use this column only to have a look at the inner workings of the package. The true values of w_i are computed one at a time as follows

```
/opts/    {nqn++;w[$2]=wa; wa*=($3+1);
           rb[$2]=$4; next}
```

On lines from the data file, print the line and get v and $\tilde{s}$. Using the *Extended Euclid Algorithm* as in [K], [C], solve the equation

$$v\bar{v} + km = 1,$$

(it gives $\bar{v}$ in u1)

```
{print $0; # print entire record
u1=1; u3=$1; v1=0; v3=key;
while(v3!=0) {
  q=int(u3/v3);
  t1=u1-v1*q;
  t3=u3-v3*q;
  u1=v1; u3=v3;
  v1=t1; v3=t3;
}
if(u1<0) u1+=key;
```

The instructions $t_i \leftarrow u_i - qv_i$ are the critical ones: If q and v_i were large at some step, the result could be wrong.

We have now to compute $s = \bar{v}\tilde{s} \bmod k$ where $\tilde{s}$ is in column 2. It is not possible to do it by straight multiplication since both $\bar{v}$ and $\tilde{s}$ are large, so we proceed by converting $\bar{v}$ to binary radix and converting multiplications to iterative modular sums in order to give the plain sum s in s:

```
for (i=0; u1>0; u1=int(u1/2))
 bb[i++]=u1%2;
sp=($2/df)%key;
for (s=0;i>0;) {
 s=(s+s)%key;
 if (bb[--i]==1) s=(sp+s)%key;
}
```

Now the quotients of iterate division by $w_1, \ldots, w_n$ will give the exam answers $b_1, \ldots, b_n$.

```
for(i=nqn;i>0;i--) {
 b[i]=int(s/w[i]); # answer of qtn i
 s%=w[i];
}
```

Comparing b_i with the correct answer gives the grade. Note that a null quotient means an unanswered question.

```
for(i=1;i<=nqn;i++) printf "%d ", b[i];
print "";         # new line
nra=0; nwa=0;
for(i=1;i<=nqn;i++) {
 if (b[i]==0)     # No answer
  c=" ";
 else
  if (b[i]==rb[i]) {
   c="+"; nra++  # Right
  } else {
   c="-"; nwa++  # Wrong
  }
 printf "%s ", c;
 }
print "";         # new line
print nra,"right,",nwa,"wrong.";
}
```

Customizing

Style commands are grouped near the end of the macro file. They include

- \headline: the headline of each page in production mode;
- \pheadline: the headline in plain mode;
- \footline and \pfootline: the same for footlines;
- \qtnprompt: material to be put in front of question in plain mode. Usually the question number;
- \anwprompt expands to material that goes before each option in production mode. Uses \labelno which delivers the suitable label number;
- \df: the detection factor d. Fine detection factors are 7 or 13 (but not 4, 3 or 11) which detect a lot of mistypings and transpositions;
- \preanwskip must expand to a vertical skip between question and first option;
- \qtnskip must expand to vertical skip between questions.

Choosing numbers. Since w_i's grow exponentially (see (2)), we are usually forced to take k very large but bounded by 2^{31} (the greatest TEX integer), thus limiting the number of questions per page. For example, no more than 19 true/false questions fit on a page. On the other hand, taking k near that upper bound will cause us trouble when used in combination with the `\stepversion` command. A rule of thumb is: The product of the stepper t by the key should already be a TEX integer, namely $tk < 2^{31}$.

Besides k, we have to choose $1 < v < k$ coprime with k. This is easy to manage simply by trying v at random: In case v is not coprime with k, we will get an

```
Invalid stepper/version number
```

error displayed on TEXing, so we have to try again. I recommend to take v large, for otherwise the answer labels of the initial questions appear in increasing sequence, thus providing students with unwanted clues on the setting and the key. A few experiments will quickly familiarize us with the right procedure.

A way to get a good v at first try consists on taking k free from low prime factors: From [S] we see that, given k, the ratio of valid v's is $\phi(k)/k$ where $\phi(k)$ stands for the Euler function. This ratio depends mainly on $1 - 1/p$ where p is the least prime factor of k. We can get both k and v coprime and large by taking prime numbers[3] and multiplying them until we have a sufficiently large k. Use the same procedure for v, but now take care to not choose any of the primes used in k.

Final notes and hints.

- Bring a portable computer at the examination room and grade exams *ipso facto*.
- Collect totals in a memory pocket calculator and upload to the main computer later.
- If your students have e-mail access, take the exam and provide each student with a control number depending on the total he gives. Ask them to send you a message with the version number, total and control. Process your mail box to give a list of grades or write a program for mailing back the grade to the originator.
- Using the `dviconcat` utility in mass production can save you a lot of work. Instead of making a bunch of `dvi` files, use that tool to concatenate all the versions into a big `dvi`. This way the printer driver has to initialize only once.

References

[C] Henri Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics. Springer (1993).

[D] Don De Smet, *TEX Macros for Producing Multiple-Choice Tests*, *TUGboat* **12**, 2 (1991).

[K] D. E. Knuth, *Seminumerical Algorithms*, Addison-Wesley (1981).

[M] Hans van der Meer, *Random Bit Generator in TEX*, *TUGboat* **15**, 1 (1994).

[Ri1] P. Ribenboim, *The Book of Prime Number Records*, Springer.

[Ri2] P. Ribenboim, *The Little Book of Big Primes*, Springer.

[Ro] Kenneth H. Rosen, *Elementary Number Theory and its Applications*, Addison-Wesley (1993).

[S] Manfred R. Schroeder, *Number Theory in Science and Communication*, Springer (1990).

◇ Jordi Saludes
Edifici de l'ETSEIT (TR5)
Colom, 11
08222 Terrassa, Spain
saludes@grec.upc.es

3 See [Ri1] or [Ri2] for a table of such numbers.

Appendix A: Example

A test **April 1995**

Name: ....

Each question has only one right answer. For each page, add the numbers at the left side of your answers and write the total by the Σ. Divide the total by 7. If your addition is right, the quotient must be integer (But this does not mean that your choices were correct). Do not forget to write down your name, but do not write anything else in the exam sheet.

Boolean question

2097123 ◇ True;
4194246 ◇ False

This question has a ‘fixed’ option at the end.

4295536 ◇ Not so well;
2147768 ◇ This is the right option;
6443304 ◇ Clearly wrong;
8591072 ◇ None of the above.

Example question with a displayed equation

$$\int_{\partial S} \omega = \int_S d\omega.$$

1610826 ◇ Option f;
1342355 ◇ Option e;
805413 ◇ Option c;
536942 ◇ Option b (and right);
268471 ◇ Option a;
1073884 ◇ Option d;
1879297 ◇ Option g;

Fuzzy question

6291369 ◇ Quite true;
3941021 ◇ Quite false;

Is this the last one?

1590673 ◇ Who knows;
3181346 ◇ Choose me;
4772019 ◇ None of the above;
6362692 ◇ All of the above.

Page 1. Version 38353. $\Sigma =$ $\Sigma/7 =$

Appendix B: File foo.tex

```
\input knst.tex
\date{April 1995}
\title{A test}
\verno=38353\SRset{162521}
%%\plainversion
\test{1234531}
\qtn Example question with
a displayed equation
$$\int_{\partial S}\omega=
  \int_S d\omega.$$
 \anw Option a;
 \Anw Option b (and right);
 \anw Option c;
 \anw Option d;
 \anw Option e;
 \anw Option f;
 \anw Option g;
\endqtn

\qtn This question has
a `fixed' option at the end.
 \Anw This is the right option;
 \anw Not so well;
 \anw Clearly wrong;
 \fix\anw None of the above.
\endqtn

\qtn Boolean question
 \anw True;
 \Anw False
\endqtn

\qtn Fuzzy question
 \Anw Quite true;
 \anw Quite false;
\endqtn

\qtn Is this the last one?
 \anw Who knows;
 \anw Choose me;
 \fix\Anw None of the above;
 \anw All of the above.
\endqtn
%\stepversion{1243}
\endtest
```

Appendix C: The macros

```
%%% knapsack test macros.
\catcode`@=11
%% Shift Register
%% from H. van der Meer, TUB 15, 1
\newcount\@SR
\def\@SRconst{2097152}
\def\SRset#1{\global\@SR#1\relax}%
\def\@SRadvance{\bgroup
 \ifnum\@SR<\@SRconst\relax
  \@A=0
 \else\@A=1\fi
 \ifodd\@SR
  \advance\@A by1 \fi
 \global\divide\@SR by2
 \ifodd\@A
  \global\advance\@SR\@SRconst\relax\fi
 \egroup}

%% Arithmetic
\newcount\@A
\newcount\@B
\newcount\@C
\newcount\@x \@x=0\relax
\newcount\@y \@y=0\relax
\newcount\@w
\newcount\key
\newcount\df \df=1\relax
\def\qtnno{\the\@x\relax}
\def\inc@#1{\advance#1 by1\relax}
\def\mod@A#1{\@B=\@A \divide\@B by#1
 \multiply\@B by#1
 \advance\@A by-\@B\relax}
\def\gcd#1#2{{\@A=#1 \@C=#2
 \loop\mod@A\@C
  \ifnum\@A>0
  \@B=\@C \@C=\@A \@A=\@B
 \repeat
 \ifnum\@C=1\else
  \errmessage{Invalid stepper/version
    number "#1"}\fi}}
\def\adv@w{{\inc@\@y
 \global\multiply\@w by \@y}}
\def\labelno{{\@A=\@B
 \mod@A\key\multiply\@A by\df \the\@A}}

%% Boxing and unboxing
\newtoks\pfootline
\newtoks\pheadline
\newbox\lqtn@bx
\newbox\lanw@bx
\newbox\canw@bx
\newbox\cqtn@bx
\def\new@page{\unvbox\lqtn@bx
 \vfill\penalty-10000
 \global\@w=\verno \adv@w}
\def\append@bx#1#2{%
 \setbox#1=\vbox{\unvbox#1 #2}}
\def\prepend@bx#1#2{%
 \setbox#1=\vbox{#2\unvbox#1}}
\def\add@anw{\@SRadvance
 \iffix@anw
  \global\append@bx{\lanw@bx}{\theanw}%
 \else
  \ifodd\@SR
   \global\prepend@bx{\lanw@bx}{\theanw}%
  \else
   \global\append@bx{\lanw@bx}{\theanw}%
  \fi\fi}
```

```
\def\add@qtn{\@SRadvance
 \iffix@qtn
  \global
   \append@bx{\lqtn@bx}{\unvbox
     \cqtn@bx}%
 \else
  \ifodd\@SR
   \global
    \prepend@bx{\lqtn@bx}{\unvbox
      \cqtn@bx}%
  \else
   \global
    \append@bx{\lqtn@bx}{\unvbox
      \cqtn@bx}%
  \fi\fi}
\def\fix{\global\fix@anwtrue\relax}
\def\fixqtn{\global\fix@qtntrue\relax}
\def\qtn{\global\inc@\@x
 \@y=0 \@B=0
 \ifplain\else
  \ifnum\@x>\botqtn
   \new@page\@next\fi\fi
 \rightanw=0
 \global\fix@anwfalse
 \setbox\cqtn@bx=\vbox\bgroup
  \noindent\qtn@pr}
\def\endqtn{\egroup\add@anw
 \ifplain\ifnum\rightanw=0
  \immediate\write16{There is no
   option marked in question
   \the\@x}\fi\fi
 \setbox\cqtn@bx=\vbox{\box\cqtn@bx
  \preanwskip\box\lanw@bx
  \qtnskip}%
 \adv@w\wr@anw
 \ifplain
  \ifnum\@w>\key
   \message{*}\new@page\fi
 \else
  {\@A=\@w \mod@A\key \global\@w=\@A}\fi
 \mark{\noexpand
  \themark{\the\@x}{\the\@w}}
 \message{.}\add@qtn}
\def\Anw{\ifnum\rightanw>0
  \errmessage{There are several
  options marked in question
  \the\@x}\fi
 \global\rightanw=\@y
 \global\inc@\rightanw
 \anw}
\def\anw{\egroup
 \ifnum\@y=0\else
  \add@anw\fi
 \advance\@B by\@w\relax\inc@\@y
 \setbox\canw@bx=\vtop\bgroup
  \relax\noindent}
\newif\ifplain
\plainfalse
\newcount\verno \verno=0
```

```
\newcount\rightanw
\newif\iffix@anw
\newif\iffix@qtn
\fix@qtnfalse
\let\read@line=\relax
\let\@read=\relax
\def\theanw{\hbox{\vrule
 height10pt width\z@\relax
 \anwprompt\ \box\canw@bx\hfill}}
\newcount\verno
\newwrite\anwfile
\newwrite\auxfile
\newdimen\labelw
\def\test#1{\key=#1
 \setbox3=\hbox{\multiply\key by\df
  \the\key}%
 \labelw=\wd3\relax
 \ifnum\verno<1\relax
  \immediate
   \openin\auxfile=\jobname.aux\relax
  \read\auxfile to\@read \@read
  \immediate\closein\auxfile\relax\fi
 \gcd{\the\verno}{\the\key}
 \@SRadvance
 \global\@w=\verno
 \header
 \setbox\lqtn@bx=\vbox{}
 \ifplain
  \message{Plain version}\SRset{0}%
  \global\@w=1
 \else
  \message{Version \the\verno}
  \immediate
   \openin\anwfile=\jobname.ans\relax
 \fi\@next}
\def\endtest{%
 \ifplain
  \write\anwfile{\string
   \Key\space\the\key\space
   \the\df\space}%
  \closeout\anwfile
 \else
  \closein\anwfile\fi
 \unvbox\lqtn@bx\vfill\supereject\end}
\def\stepversion#1{\gcd{#1}{\the\key}%
 \immediate
  \openout\auxfile=\jobname.aux\relax
 {\ifnum\@SR=0
   \@SR=\@SRconst
   \divide\@SR by3 \fi
  \@SRadvance
  \@A=\verno
  \multiply\@A by#1\relax
  \mod@A\key\relax
  \immediate\write\auxfile{\string
   \verno=\the\@A\space
   \string\SRset{\the\@SR}}}%
 \immediate\closeout\auxfile}
\def\plainversion{\plaintrue\verno=1
```

```
 \let\newpage=\new@page
 \def\wr@anw{\immediate
   \write\anwfile{\string\opts
    \space\the\@x\space\the\@y\space
    \the\rightanw\space\the\@w\space}}
 \immediate
  \openout\anwfile=\jobname.ans\relax
 \def\qtn@pr{\llap{\qtnprompt}}
\def\anwprompt{\hbox to\labelw{\hss
 \ifnum\rightanw=\@y
  $\bullet$\ \fi}}
\footline=\pfootline
\headline=\pheadline
\let\@next=\relax
\output={\botmark\plainoutput}
\def\add@qtn{\unvbox\cqtn@bx\penalty-50}}
\def\themark#1#2{\immediate
 \write\anwfile{\string\pagebot
  \space #1\space}%
 \global\divide\@w by#2\relax}
\let\wr@anw=\relax
\let\qtn@pr=\relax
\def\@next{\read\anwfile to\@read
 \@read}
\def\Key #1 #2 {\@next}
\def\opts #1 #2 #3 #4 {\@next}
\def\pagebot #1 {\def\botqtn{#1}}
\def\botqtn{1000}
\let\newpage=\relax

%% Style commands
\def\qtnprompt{\bf\qtnno.\ }
\pfootline={\hss
 Plain version $\dots$\hss}
\pheadline={\ifnum\folio>1
 \thetitle\hss\folio\else\hss\fi}
\footline={Page \folio. Version
 {\tt\the\verno}.
 $\Sigma=$\hskip7em
 $\Sigma/\the\df=$\hfill}
\def\date#1{\def\thedate{#1}}
\def\title#1{\def\thetitle{#1}}
\def\header{
 \line{\bf\thetitle\hfill\thedate}%
 \smallskip
 \line{Name:\ \dotfill}%
 \medskip
{\baselineskip=9pt
 \noindent\small
 Each question has
 only one right answer.
 For each page, add the numbers at
 the left side of
 your answers and write the total by
 the $\Sigma$. Divide the total
 by~\the\df. If your addition is right,
 the quotient must be integer
 (But this does not mean
 that your choices were correct).
 Do not forget to write down your name,
 but do not write anything else in the
 exam sheet.\par}
 \medskip}
\df=7
\def\anwprompt{\hbox to\labelw{\it
 \hss\labelno\ $\diamond$}}
\def\qtnskip{\vskip 20pt plus25pt }
\let\preanwskip=\smallskip
\font\small=cmr8
```

A puzzling TEX macro

Peter Schmitt

What would you answer if I ask you to define a control sequence whose replacement text is a single left brace `{`? You would answer (probably without hesitation) that such a control sequence does not exist. How would you react if I insist, and still ask you to generate such a control sequence? You probably will produce the TEXbook, and point to a passage which explicitly states that this is impossible, or you might suspect some trick question and try to find a double meaning in my question.

But I am stubborn and repeat my challenge more precisely:

1 Problem

Define a macro which can be used to define a control sequence which expands to a single left brace `{`.

Or, more precisely:

After the application of the macro, say `\MakeBrace\brace`, the replacement text of `\brace` should consist of a single token, a character of category 1.

The following remarks should disturb all doubts about hidden meanings in the formulation of the 'puzzle':

(1) If the problem is correctly solved and if `\cs` is a macro that takes a single (undelimited) argument,
then `\expandafter\cs \brace` ⟨*argument*⟩`}`
is equivalent to `\cs {`⟨*argument*⟩`}`

(2) If `\def\Def{\def\cs}`
then
`\expandafter\Def \brace` ⟨*macro text*⟩`}`
is equivalent to `\def\cs {`⟨*macro text*⟩`}`

Moreover, I concede the following passage:

(3) According to the TEXbook (page 203), for macros "all occurrences of `{` and `}` in the ⟨*replacement text*⟩ are properly nested."

And I stress:

(4) Of course, `\def\MakeBrace {{\iffalse}\fi}` is not a solution of the problem.

Are you intrigued? Do you want to try to find out by yourself? Then be warned: It is *not* a trick question, but the solution *is* tricky, and it involves a *trick* you might consider to be unfair. (Hint: Obviously, re-reading the TEXbook is *not* likely to help you!) For all the others who are just curious and only want to check if (or where) I am cheating — here is how to do it:

2 The Solution

The solution of the problem is based on an observation (over which I stumbled quite accidentally) concerning the behaviour of the `\read` command. `\read` stores "the contents of the next line" in a control sequence `\cs`, but "additional lines are read, if necessary, until an equal number of left and right braces has been found." If the end of the file is prematurely reached, TEX complains and issues an error message (`File ended within read`). However, quite surprisingly, the input read so far is available in the control sequence `\cs` which therefore contains an unmatched left brace `{`.

The definition of `\MakeBrace` takes advantage of this behaviour. First it generates a file containing a single brace `{` only. (Of course, this file could also be prepared manually.) Then the file is opened for input and (using `\read to#1`) it is read to the control sequence to be defined. Before that, the `\endlinechar` is set to `-1`, i.e., it is removed, since otherwise an additional `\par` would be read. Moreover, in order to hide the error message, the macro temporarily switches to `\batchmode`.

```
\def\MakeBrace #1{\bgroup \batchmode
%        % supress error message
  \immediate\openout1 brace
    \escapechar-1
    \immediate\write1{\string\{}
%        % write a single {
    \immediate\closeout1
  \immediate\openin1 brace
    \endlinechar-1 \global\read1 to #1
%        % (incomplete) read of {
    \immediate\closein1
                 \egroup \errorstopmode }
%        % return to normal
% test:
\MakeBrace\brace  \show\brace
\message{.\brace.}}
%                    % note second } !
```

3 Remark

Attempting to apply the same idea to a right brace fails: When `\read` encounters an unmatched brace `}` TEX does not even bother to stop but (silently!) discards the rest of the line, including the offending brace. Consequently, only properly nested braces are read, and the unmatched brace has the same effect as a comment character.

4 Summary

Reading (by a `\read` command) from a file which is not balanced with respect to `{` and `}` causes TEX to behave in a surprising way which is not documented in the TEXbook. In one case, an unmatched `}` is not reported, in the other case, an unmatched `{` produces a macro which — according to the rules — cannot exist. Does this constitute a bug or a feature of TEX?

◇ Peter Schmitt
Institut für Mathematik
Universität Wien
Strudlhofgasse 4
A-1090 Wien, Austria
schmitt@awirap.bitnet
schmitt@pap.univie.ac.at

LaTeX

A LaTeX Tour, part 2: The Tools and Graphics distributions

David Carlisle

Introduction

In the previous article in this series I started by giving a description of the files in the 'base' LaTeX distribution. In part 2, I shall cover the 'tools' and 'graphics' distributions. These are distributed in the `tools` and `graphics` subdirectories of the CTAN directory `macros/latex/packages`. Although these files are not part of the minimal base distribution they should normally be included in the LaTeX installation at any site. The LaTeX book assumes that at least the graphics distribution is installed.

The primary source for LaTeX is the 'CTAN'[1] network of archives, so if I refer to path names of files this relates to the CTAN file structure. Note however that if you obtained LaTeX as part of a 'pre-packaged' TeX distribution, then these files may have been moved (typically documentation files may be separated from TeX source files). I hope this will not cause any confusion.

The Tools Distribution

The tools distribution consists of packages written by individual members of the LaTeX3 project. They are supported by the same mechanism as the base LaTeX distribution, that is, any problems should be reported using `latexbug.tex` and the LaTeX bug report database, as described in part 1. Note that this bug report system should *not* be used for 'contributed' packages that one may find in the `macros/latex/contrib` area of the CTAN archives.

Packages Extending the array and tabular Environments The first group of packages extends the functionality of the standard LaTeX array and tabular environments. These are all described in Chapter 5 of *The LaTeX Companion*, as well of course as in the source '`.dtx`' files which may be processed by LaTeX to produce typeset documentation, and optionally code listings.

`array` Extended versions of the array, tabular and tabular* environments. The principal advantage of the versions provided by this package is that you can specify typesetting instructions to apply to a whole column of the table. As well as the usual `clr` column specifiers, one may add commands at the beginning of each entry with > and at the end of each entry with <. So a column specifier of `>{\bfseries}c` would produce a bold, centred column of a table.

The array package also provides a command `\newcolumntype` for defining new column specifiers, in addition to the standard ones. This is used by some of the packages described below.

`dcolumn` Alignment on 'decimal points' in tabular entries. Requires array. This package provides a new column specifier `D` which may be used to produce columns of numbers aligned on a decimal point '.' or some other symbol, such as '$\cdot$' or ','.

`delarray` This package requires the array package. It provides a mechanism for specifying 'large delimiters' around arrays. This is most convenient for putting brackets around arrays that are to be aligned on their top or bottom row (when the 'obvious' construction with `\left` and `\right` does not work). Compare the standard

```
\left(
\begin{array}[t]{c}a\\b\end{array}
\right)
\left(
\begin{array}{cc}a&b\end{array}
\right)
```

$$\left(\begin{array}[t]{c}a\\b\end{array}\right)\left(\begin{array}{cc}a&b\end{array}\right)$$

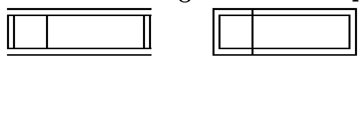
with the effect produced using the delarray syntax.

```
\begin{array}[t]({c})a\\b\end{array}
\begin{array}({cc})a&b\end{array}
```

$$\begin{array}[t]{c}\left(\begin{array}{c}a\\b\end{array}\right)\end{array}\left(\begin{array}{cc}a&b\end{array}\right)$$

`hhline` Finer control over horizontal rules in tables. Requires array. Standard LaTeX's `\hline\hline` construction produces a double rule across a table, but the user has no control over how this rule interacts with vertical rules. Using the `\hhline` command provided by this package gives 'corners' where a double horizontal rule meets a double vertical rule, and other similar effects.

Compare the first, standard, construction with the following `\hhline` sample:



[1] `ftp.tex.ac.uk` in the UK, `ftp.dante.de` in Germany, and `ftp.cdrom.com` in the US

`longtable` Standard LaTeX tables (i.e., the tabular environment) produce 'boxes' that cannot be broken across a page. This has advantages in that the table can then be positioned just like a large 'character' (say centred by the center environment), but has the disadvantage that large tables need to be broken by hand to fit on the page. The longtable environment is essentially the same as tabular but produces tables that break at page boundaries, and has some additional commands to control 'head' and 'foot' lines of the table that are added to each page. If the array package is also loaded, then the extra features may also be used in longtable column specifications. Note that longtable can deal with *very* long tables, longer than can be stored in memory by TeX's primitive `\halign` command.

The longtable package has a few quirks and features that mean that it is not suitable in all cases. An alternative package (currently maintained by Johannes Braams, but as a contributed package, not as part of the tools distribution) is the package supertab which provides a similar supertabular environment.

`tabularx` Defines the tabularx environment which is similar to tabular* but modifies column widths, not inter-column space, to achieve a desired table width.

One common request is to combine the features of tabularx with longtable, i.e., have a table across multiple pages, in which the widths of the 'parbox' columns are calculated automatically. This functionality is not provided by the standard packages in the tools distribution, but the experimental contributed package ltxtable does provide such an environment. (ltxtable is written by the same author as longtable and tabularx; however, problems with ltxtable should *not* be addressed to the LaTeX bugs system.) An alternative to ltxtable is Anil Goel's contributed package, ltablex, a similar merger which is simpler to use than ltxtable, but not quite as powerful.

Missing File Error Control Files Although these files (which are all generated from the same source file, `fileerr.dtx`) are distributed as part of the LaTeX distribution, they are possibly of more use when used with *other* formats. The primitive TeX behaviour if asked to input a non-existent file is to offer a prompt:

```
Please type another input file name:
```

You *must* type a valid file name to this prompt or TeX just repeats the request. On some systems you can use a mechanism to abort the job (e.g., control-C or control-Z) but there is no way to tell TeX to skip the input or do any other error recovery.

To avoid this unpleasant loop, LaTeX always checks that a file exists before trying to input it (unless you use the primitive `\input filename` syntax with no `{ }` around the argument).

If you do encounter this loop using a different format, or with LaTeX, by mistyping the file name on the command line, then the following `.tex` files provide valid filenames that you can easily remember to type at the missing file prompt. The actions that each of these `.tex` files takes is designed to mimic the actions that are possible after a TeX error.

`h` Typing `h` ⟨*return*⟩ to the missing file prompt will cause TeX to input `h.tex`. This produces a helpful message, and then produces the normal 'error prompt' i.e., `?` so you can hit ⟨*return*⟩ to move on, or `x` to quit, or whatever.

`s` Typing `s` ⟨*return*⟩ inputs `s.tex` which puts TeX into 'scroll mode'. This means that it will scroll past future errors without stopping.

`x` Typing `x` ⟨*return*⟩ causes the current TeX run to be aborted.

`e` The file `e.tex` is in fact the same as `x.tex` but allows `e` to be given as an answer to the missing file prompt similar to the `e` response to the error prompt (which is supposed to start up an editor but usually is the same as `x`).

␣ If the operating system allows there will also be a file `.tex` which does nothing, this will mean that just hitting ⟨*return*⟩ in response to the missing file prompt inputs `.tex` and allows TeX to proceed with the original file. Some operating systems object to a file with only an extension and no filename before the '`.`' so this option may not be available to you. Most TeX distributions include a file `null.tex` which is also empty, so if you do not have the option of installing the file `.tex` you may type `null` ⟨*return*⟩ in response to the missing file prompt, which will also allow TeX to proceed.

Miscellaneous Tools Packages

`afterpage` Defines an `\afterpage` command that saves up its argument and executes it after the current page (i.e., at the top of the next page). LaTeX's output routine was *not* designed with the idea that packages might want to play this kind of trick, so this package is particularly fragile. In fact it was only written as a kind of private joke; I noticed the comment "*Output routines are always protected by enclosing them in groups, so that they do not inadvertently mess up the rest of TEX*" in the TEXbook, and wanted to answer[2] the question, "*Where do you end up if you jump out of that group with `\aftergroup`?*" Despite this, judging by reactions, people do seem to find the package useful...

`enumerate` Extended version of the enumerate environment. The environment is given an optional argument which controls how the counter is printed. For example `\begin{enumerate}[a)]` would produce items labelled 'a)' ' b)' ' c)'.

`ftnright` Place footnotes in the right hand column in two-column mode. Normally LATEX places footnotes at the bottom of each column. This package causes the footnotes for both columns of a page to be set in the normal text area at the end of the second column on each page. It currently works only with the standard two column mechanism, not with the mechanism of the multicol package.

`indentfirst` Indent the first paragraph of sections etc. This very small package just suppresses the usual LATEX mechanism which ensures that the first paragraph of each section is not indented.

`multicol` Typeset text in columns (up to 10 columns per page), with the length of the final columns 'balanced'. *Baskerville* uses this package to balance the columns at the end of every article. Unlike the standard `\twocolumn` command, this package allows changing the number of columns part way down a page. It does have some restrictions on the use of floats which means that it is not suitable for all purposes. Also (uniquely for the files in the core LATEX distribution) this package has special conditions on commercial use.

`rawfonts` Preload fonts under the old internal font names of LATEX 2.09. Not recommended for new packages, but may help when updating old files.

`somedefs` This package is not intended to be called directly by a document, but may by used (via `\RequirePackage`) to build a package in which you want the default behaviour to be to execute *all* possible options, but that the user may execute just some of the options by specifying options in the `\usepackage` call. This is used in the rawfonts package above to allow just some of the 'old' font names to be defined rather than all of them.

`theorem` Flexible definition of 'theorem-like' environments. The standard `\newtheorem` command gives some control over the title and numbering of 'theorem-like' declarations, but is not very flexible. The theorem package provides an enhanced declaration scheme which gives control over the fonts used in the heading and theorem body, and such details as whether the numbering is '**Theorem 1**' or '**1 Theorem**'. Recently this package has acquired a close cousin, the amsthm package, part of the 'AMS-LATEX' collection. The AMS variant has perhaps slightly simpler user-syntax but is used in much the same way.

`varioref` Provides `\vref` and related commands. `\vref` is like a combination of `\ref` and `\pageref` which produces references such as 'Figure 2 on page 3'. However, it omits the page number if it is on the current page, and replaces it by phrases such as 'on the facing page' when appropriate.

`verbatim` Flexible version of verbatim environment. The standard LATEX verbatim environment can not easily be used in the definition of other environments as typically the `\end` of the newly defined environment is not recognised as such, but is treated as verbatim text. This package re-implements verbatim such that (with some restrictions) it can be used in the definition of other environments and commands. It also defines some such derived environments, for inputting and writing files verbatim, and for adding line numbers, and also a comment environment that ignores all the environment body.

`xr` The xr (external references) package allows one LATEX document to access the `.aux` file of another. So if file `fileA` has a section marked with `\label{xyz}` then file `fileB` may refer to that section using `\ref{xyz}` just as if it were part of the same document. This requires the file `fileA.aux` created when `fileA` was processed to be still available when `fileB` is processed. (This package was originally by Jean-Pierre Drucbert, but was recoded and adopted into the tools distribution.)

`xspace` One of the more common errors in TEX documents is to use a command such as `\TeX` within text, but forget to follow it with `\␣` or `{}`. This package defines a command `\xspace` which may be used at the end of the definition of such a 'text command'. It looks ahead

[2] The answer incidentally is not at the top of the next page, but rather any of the places where the TEXbook uses the magic phrase "*exercises the page builder*".

and adds a space unless the next token is a punctuation character.

Packages for Drafts and Tests

`fontsmpl` Package and test file for producing 'font samples'. The base distribution contains a file `nfssfont.tex` that shows some small samples, and a character table for a given font. `fontsmpl.tex` produces a much more extensive test showing examples of all the fonts in a given family. If you want to devise your own similar test suite you may use the fontsmpl package, following the examples in `fontsmpl.tex`.

`layout` Defines a `\layout` command that produces a half size 'picture' of the document page settings such as `\textwidth`, `\oddsidemargin`, ... together with a table of their values. This is quite useful when designing a new class file, as it gives a visual representation of how the various areas of the page for headlines, body text, marginal notes, etc., relate to each other.

`showkeys` LaTeX's automatic numbering and cross referencing feature is one of its strongest points, as it makes editing a document (and thus potentially changing the numbering throughout the rest of the file) quite painless. However, one disadvantage is that when reading a printed draft, one sees 'final' numbers rather than the symbolic names that are used in the source file's `\label` and `\ref` command. This package makes these symbolic names, or 'keys', appear in the margin in the case of `\label` and `\bibitem` or raised above the number, like this $\overset{\boxed{\text{lbl}}}{1}$, in the case of `\ref` and `\cite`. Some people find the raised labels above cross references distracting and so a package option turns them off, just leaving the marginal notes showing the `\label` and `\bibitem` keys. $\boxed{\text{lbl}}$

The Graphics Distribution

TeX (and the `dvi` format) is only designed to deal with rectangular boxes consisting of text, white space or rectangular rules. However it has an 'escape mechanism', the `\special` primitive command that allows processing instructions to be passed straight from TeX (via the `dvi` file) to the 'driver' program that is used to process (e.g., preview or print) the `dvi` file. TeX places essentially no restrictions on what instructions may be passed via `\special`, and so the possibilities are unlimited...

Most modern drivers can import 'graphic' files of various sorts. Those drivers that are producing PostScript can often do more extensive manipulations of the typeset text, such as scaling or rotation of the text, or even writing text along an arbitrary curve. Many of these drivers can also support colour to some extent. Unfortunately as all these features require that the `dvi` file stores processing instructions for the driver, it means that the `dvi` file is not portable to a site that uses a different driver program. There have been many attempts over the years to coordinate the `\special` syntax used by the different drivers, so that they would all accept a common core of processing instructions, but there has been notable lack of success in such efforts to date...

As a 'next best thing' to having portability at the `dvi` level, LaTeX supplies a suite of standard graphics commands provided by the packages described in this section, so that at least TeX source files should be reasonably portable. At a given site the graphics packages will be customised to use a suitable 'back end' file that converts the LaTeX syntax into the form required by the local driver. This should mean that as long as both drivers support some feature, such as including PostScript graphics, a file just needs to be re-processed with LaTeX to use the `\special`s at the new site; the LaTeX file does not need to be edited. Although this suite of programs was devised as part of LaTeX, users of other TeX formats may use them by way of the interface available from CTAN hosts in `macros/generic/graphics`.

Documentation All the packages in this distribution are, as usual, distributed as documented sources in `dtx` form, however the documentation in these package sources is rather technical. A separate 'User Guide' is available as LaTeX source in `grfguide.tex` and also in pre-formatted form in the PostScript file `grfguide.ps`.

The color package (which produces colours despite the strange spelling) and the graphics package are also described in Lamport's LaTeX manual. An alternative to `grfguide.tex` as a free source of documentation is Keith Reckdahl's *Using EPS Graphics in LaTeX 2ε Documents* distributed from CTAN sites in the file `info/epslatex.ps` and published in *TUGboat* 17, no. 1–2. This document covers the graphicx package in some depth, and also related contributed packages for controlling figure placement and captions, and the psfrag system for overlaying LaTeX text over a PostScript diagram.

Colour

`color` Produce coloured effects in your document. The `\color{red}` declaration would make all the following text red, the similar `\textcolor` command takes an extra argument that specifies the text to be coloured (by analogy with `\rmfamily` and `\textrm`).

One may also produce boxes with coloured backgrounds using the `\colorbox` command.

Accurate treatment of colour is probably the feature that requires the most 'help' from the driver program. If your driver was not specifically written to support colour then probably the color package will not work at all, or will be limited to regions of colour that fall on one page, and all the current colours will be 'forgotten' at a page break.

`pstcol` The pstricks package of Tim van Zandt provides a very powerful interface to PostScript. Unfortunately the package has some slight incompatibilities with the color package. If a document loads pstcol, both color and pstricks are loaded, and then a few internal pstricks functions are redefined to repair the incompatibility.

Rotation, Scaling and Graphics Inclusion

`graphics` This is the core LaTeX package for rotation (`\rotatebox`), scaling (`\scalebox` and `\resizebox`) of text, and the inclusion of graphics images (`\includegraphics`). Unlike the old LaTeX 2.09 packages such as psfig the `\includegraphics` command is not restricted to PostScript graphics, but can include any graphics formats that your driver supports.

`graphicx` Often when including graphics files one needs to specify combinations of scaling and rotation and other special effects. The graphics package uses standard LaTeX 'positional' optional arguments which means that it is not practical for any command to support more than a couple of optional arguments. The graphicx package calls the graphics package internally, but offers a more powerful and friendly 'named argument' interface in which an arbitrary number of optional keys may be set in one `[ ]` argument. For instance to include a graphic scaled to half size, and rotated through 90°, one can specify

```
\includegraphics[scale=.5, angle=90]{file}
```

To do the equivalent with the graphics package would require nested calls of `\includegraphics` inside `\scalebox` inside `\rotatebox`.

`lscape` Provides a landscape environment within which the body of every page is rotated through 90°. The page head and foot are not rotated, but stay in their usual positions. It requires the graphics package which is used to handle the rotation.

`epsfig` The obsolete LaTeX 2.09 did not come with a standard graphics package. Two popular contributed packages to include PostScript graphics were psfig (T. Darrell) and epsf (T. Rokicki). Sebastian Rahtz merged and extended these to produce the package epsfig. The epsfig package became very popular, especially after it was given extensive coverage in *The LaTeX Companion*. For this reason the current distribution contains a package called epsfig so that old documents do not need converting to the new system. However this epsfig is just a wrapper that converts the old syntax into calls to the new `\includegraphics` command and so should not now be used for new documents.

Driver Files As mentioned above, these packages all require customisation to a particular driver. This may be specified either in a site configuration file, or as a package option in the document. The code for these drivers is all stored in '`.def`' files, so for instance the code for the *dvips* driver (and also for *xdvi* which uses the same `\special` syntax) is stored in `dvips.def`. All these driver files are derived from the same source, `drivers.dtx`, except for the Textures file which is currently distributed as a separate `textures.def` contributed by Arthur Ogawa. One special `.def` file does not correspond to a driver, `dvipsnam.def` predefines the 60+ colours that are 'known' to the *dvips* driver. It may be used with other drivers as well, as described in the color package documentation.

Other Graphics Packages The remaining two packages do not have code that is specific to dvi driver programs, and so in some sense do not really belong in the graphics distribution; they are used by the graphics and graphicx packages. In fact the code of either of these packages may be extracted and used in any format based on plain TeX. They do not use any LaTeX specific features.

`keyval` The graphicx package makes use of a 'named argument' or 'key equals value' syntax as described above. Keyval provides a general parser for such a syntax, so this package is unlikely to be directly called within a document, but may be loaded by `\RequirePackage` by any package or class file that needs to define commands with such a syntax.

`trig` This package provides functions for calculating the trigonometric functions sin, cos and tan. These are used by the graphics package for determining the amount of space a rotated box will take up.

Coming Soon

Part 3 of this tour will describe the files of Johannes Braams' babel distribution of packages for multilingual typesetting, the psnfss distribution of PostScript font related packages, and mfnfss distribution of packages for loading 'Pandora' and 'Old German' fonts.

◇ David Carlisle
Mathematics Department
Manchester University
Oxford Road
Manchester, England M13 9PL UK
`carlisle@ma.man.ac.uk`

Late-Breaking News

Production Notes

Mimi Burbank

This issue is more than two months late and the team wishes to apologize to our readers. Illness and work demands have prevented *all* of us from spending the necessary time on production (It must be true that troubles occur in 'threes'.).

Several articles from the TUG'96 conference are included in this issue — those articles by Petr Sojka (page 244), Sergey Lesenko (page 252), Irina Makhovaya (page 259), and Kees van der Laan (page 269).

All files were received electronically. 85% of the articles received were in LaTeX 2_ε, with only three articles being received in plain TeX. Most articles were received as fully tagged for *TUGboat*, using either `plain`-based or LaTeX conventions described in the Authors' Guide (see *TUGboat* 10, no. 3, pages 378–385).

Output The final camera copy was prepared at SCRI on an IBM RS6000 running AIX, using the *Web2C* implementation of TeX.

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

Future Issues

The next issue will be a "theme" issue, on "non-mathematical" TeX, TeX and the humanities, TeX and the world's languages, and Pierre MacKay and Christina Thiele are the guest editors.

Suggestions are welcome for prospective topics and guest editors. Send them to the Editor, Barbara Beeton (see address on page 239), or via electronic mail to `TUGboat@ams.org`.

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

Call for Papers

TUG'97

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 center is 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 will be available to conference attendees.

The theme of the meeting will be "*TEX Comes Home*". We are welcoming papers on any topic you wish to pursue, including TEX for beginners, how to improve your TEXniques, TEX in industry and commerce, book building with TEX, extensions to TEX, and TEX on the Internet.

Please forward your abstracts or questions to `tug97@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

Submission of Abstracts	Jan 31, 1997
Notification of acceptance to authors	Feb 14, 1997
Preliminary Papers Due	Apr 2, 1997
Preprint Deadline	May 15, 1997
Camera Ready Copy Deadline	June 15, 1997

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.