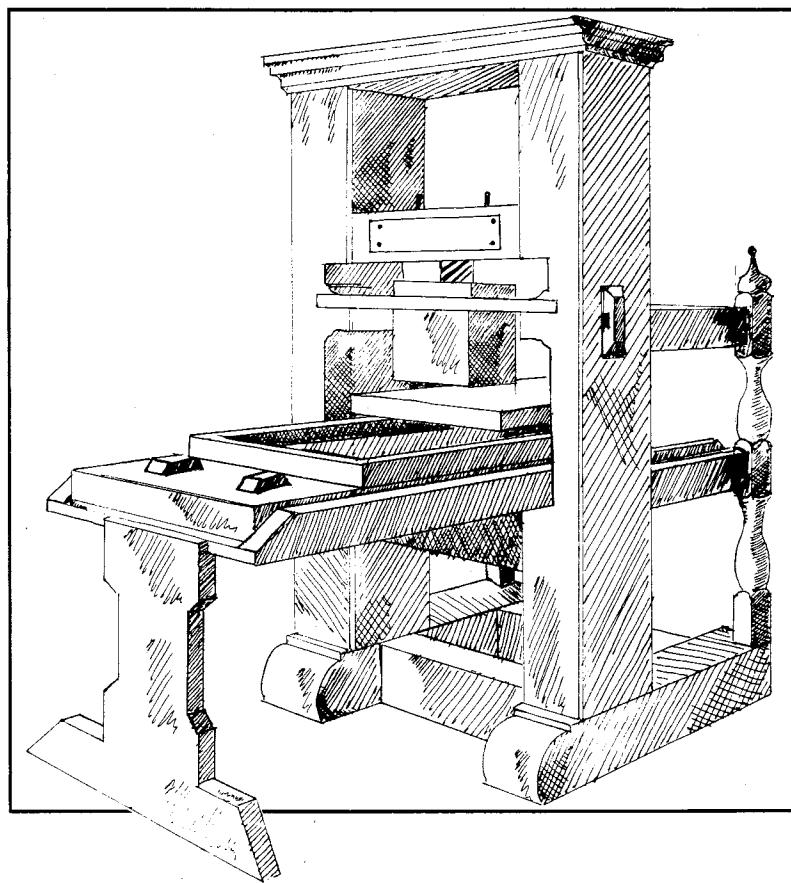


TUGBOAT

The Communications of the TEX Users Group



Volume 14, Number 1, April 1993

TeX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published four times a year plus one supplement by the TeX Users Group, Balboa Building, Room 307, 735 State Street, Santa Barbara, CA 93101, U.S.A.

1993 dues for individual members are as follows:

- Ordinary members: \$60
- Students: \$30

Membership in the TeX Users Group is for the calendar year, and includes all issues of *TUGboat* and *TeX* and *TUG NEWS* 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 85.

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: North America \$60 a year; all other countries, delivery by surface mail \$60, by air mail \$80.

Second-class postage paid at Santa Barbara, CA, and additional mailing offices. Postmaster: Send address changes to the TeX Users Group, P.O. Box 869, Santa Barbara, CA 93102-0869, 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.

TUGboat © Copyright 1993, TeX Users Group

Permission is granted to make and distribute verbatim copies of this publication or of individual items from this publication provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this publication or of individual items from this publication under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this publication or of individual items from this publication into another language, under the above conditions for modified versions, except that this permission notice may be included in translations approved by the TeX Users Group instead of in the original English.

Some individual authors may wish to retain traditional copyright rights to their own articles. Such articles can be identified by the presence of a copyright notice thereon.

Board of Directors

Donald Knuth, *Grand Wizard of TeX-arcana*[†]

Christina Thiele, *President*^{*}

Ken Dreyhaupt^{*}, *Vice President*

Bill Woolf^{*}, *Treasurer*

Peter Flynn^{*}, *Secretary*

Peter Abbott, *Special Director for UKTeXUG*

Alain Cousquer, *Special Director for GUTenberg*

Roswitha Graham, *Special Director for the Nordic countries*

Kees van der Laan, *Special Director for NTG*

Joachim Lammarsch, *Special Director for DANTE*

Barbara Beeton

Luzia Dietsche

Michael Ferguson

Raymond Goucher, *Founding Executive Director*[†]

Yannis Haralambous

Doug Henderson

Alan Hoenig

Anita Hoover

Mimi Jett

David Kellerman

Nico Poppelier

Jon Radel

Hermann Zapf, *Wizard of Fonts*[†]

*member of executive committee

†honorary

Addresses

General correspondence: 805-963-1338

TeX Users Group

P. O. Box 869

Santa Barbara, CA 93102-

0869 USA

Payments:

TeX Users Group

P. O. Box 21041

Santa Barbara,

CA 93121-1041 USA

Parcel post,

delivery services:

TeX Users Group

Balboa Building

Room 307

735 State Street

Santa Barbara, CA 93101

USA

Telephone

805-963-1338

Fax

805-963-8358

Electronic Mail

(Internet)

General correspondence:

TUG@Math.AMS.org

Submissions to *TUGboat*:

TUGboat@Math.AMS.org

TeX is a trademark of the American Mathematical Society.

With type as with philosophy, music and food, it is better
to have a little of the best than to be swamped with the
derivative, the careless, the routine.

Robert Bringhurst

The Elements of Typographic Style
(1992)

TUGBOAT

COMMUNICATIONS OF THE **TEX** USERS GROUP
EDITOR BARBARA BEETON

VOLUME 14, NUMBER 1

PROVIDENCE

• RHODE ISLAND

APRIL 1993

• U.S.A.

TUGboat

During 1993, the communications of the TeX Users Group will be published in four issues. One issue (Vol. 14, No. 3) will contain the Proceedings of the 1993 TUG Annual Meeting.

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

Submissions to *TUGboat* are reviewed by volunteers and checked by the Editor before publication. However, the authors are still assumed to be the 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. 14, No. 1; deadlines for that issue will have passed by the time this issue is mailed. Deadlines for Vol. 14, No. 2 are February 16, 1993, for technical items, and March 16, 1993, for reports and similar items. Mailing dates for these two issues are scheduled for March and May. Deadlines for future issues are listed in the Calendar, page 77.

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. 3).

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

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

Reviewers

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to **TUGboat@Math.AMS.com** or to the Editor, Barbara Beeton (see address on p. 3).

TUGboat Editorial Board

Barbara Beeton, *Editor*

Victor Eijkhout, *Associate Editor, Macros*

Jackie Damrau, *Associate Editor, LATEX*

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

See page 3 for addresses.

Other TUG Publications

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

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

APS μ5 is a trademark of Autologic, Inc.

DOS and MS/DOS are trademarks of MicroSoft Corporation

METAFONT is a trademark of Addison-Wesley Inc.

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

PostScript is a trademark of Adobe Systems, Inc.

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

Textures is a trademark of Blue Sky Research.

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

General Delivery

Opening words

Christina Thiele

Hello and welcome to you all — hello to those of you renewing your memberships, and welcome to those of you who are becoming TUG members for the first time. You arrive at an interesting time — but then, there's always something interesting going on!

This time, it's a new year with a new President, and a new Executive Director. It's a year where the office has moved from the east coast to the west. It's a year where we'll see our first annual meeting located outside North America.

In the last issue of *TUGboat*, Malcolm Clark, the outgoing president, warned against expecting that the new president, and new executive director, be seen as magic bullets ... Speaking for myself, I just don't move that fast, and as for magic, well, I think solid effort and involvement are going to be more effective in the long haul.

Perhaps a bit of background first ... I've been a TUG member since 1986, and attended my first meeting the following year, in Seattle, where I also gave a paper on my own proud beginnings with *TeX* ... the audience was very kind that year. Like many of you, I had to work things out for myself, while avoiding learning things that I didn't think I'd need (!), and was pleased as punch when a file only tripped five times over errors.

Following that first encounter with other like-minded *TeX* users at the University of Washington, I became a member of TUG's board in 1988, served on the program committee for four years, was editor of our proceedings for three years (writing the first guidelines, and then promptly overhauling them each year, or so it seemed), and finally felt I could retire from conference activities after 1991. Besides, I'd acquired a new job: editor of the fledgling newsletter *TeX* and *TUG NEWS*.

And now I have arrived here, as TUG's president. Along with Pat Monohon, our Executive Director, I hope to inject new life and interest — and fun — into TUG.

We are now in our 14th year. That's quite an age, in a sense. It means we have a history (more on that as the year progresses ...).

I used to carve that history up into pieces: "the days of the implementors" — site coordinators, people who were working hard to get *TeX* up and

running on this or that machine; followed by "the days of the users" — *TeX* applications to real everyday situations on the job, in the production office, around the world via e-mail and ftp.

But ... have we really left the days of developers behind? Look at the work being done on *LATEX3* and the New Font Selection Scheme; the discussions, both heated and reasoned, on changing *TeX*; the advent of DC and virtual fonts. Development has taken on a new face, that's all. In a sense, we're renewing, reinventing ourselves. Revitalising *TeX*, revamping the way we use it, learning to be a bigger player in the electronic field.

We're 14 years old — what *TeX* will eventually turn into is being shaped right now. There's almost too much happening! How can we, the users, keep up? I myself only just got a PostScript printer. I'm about to embark on using non-CM fonts. This is a mental lurch for me: I'm not really in a position to say to someone else: "Look. You set it up for me. I'll just do the encoding as usual — but you take care of the *TeXnical* details." So for me, I'm extremely grateful that I belong to TUG — as a user, this is where I know I can look for help, for support, ... for solutions.

And that's where I think TUG has its greatest role to play: in serving its members (in serving all *TeX* users, really). Providing information, helping make contacts, finding answers. Not just via publications such as *TUGboat* or *TTN*, but directly from fellow members.

If you're a renewed member, go look in last year's membership directory (new members — you'll be getting a treasure trove of information when the directory arrives this summer). Don't look at the alphabetical listing of members — go to the Listing by Institutions and see who else at your university or company is a TUG member. Amazing, isn't it! Better yet — go look at the section which lists TUG members by their city and country. Did you know we have someone from the Canary Islands?! Saudi Arabia?! Look at how many members there are in Sweden and Japan, and at Los Alamos Laboratories in New Mexico! Amazing. Thinly spread around the globe, granted. But no matter where you are, there's a TUG member playing or working with *TeX*.

And yet ... we know that the number of people actively using *TeX* is an even greater horde — invisible to us, but facing exactly the same problems, enjoying exactly the same thrills of accomplishment.

TeX is everywhere. TUG isn't ... yet! But we are becoming more and more the focal point for information on *TeX* and its relations. That is one of my goals, and one shared by Pat — to work on ways

of improving our ability and our capacity to provide and distribute information. We have had recent inquiries from people involved in developing software which hopes to address many users, including those working with TeX. What is so encouraging is not only that they're making sure to include TeX, but that they are also coming to TUG to signal their work, and ask for comments and suggestions. We must continue to extend our presence, TeX's presence, into adjacent development work. If we aren't part of new product solutions, we'll be left stranded. Something to keep in mind as we approach the mid-point in our second decade.

-- * --

And now is perhaps the moment to introduce the new office to you. Santa Barbara is a long way from Providence, and moving a lot of boxes (350 or more, at last count) has proven to be quite an operation. But everyone—and everything—is more or less settled in now, and things should be in full gear by mid-Spring.

TUG's offices (there are three rooms) are on the 3rd floor of a 6-storey restored building at 735 State Street, Santa Barbara. Other tenants in the building include Mission Research Corp. and Condor Engineering, both of whom are TeX-aware. Nice company, indeed!

Pat Monohon has been a member of the TeX Users Group since 1984, with several years' experience working in various European locations. She has virtually taken up residence in the offices—and has given more than one international caller a shock when they connected with her rather than the answering machine.

Along with Pat, there are two full-timers in the office: Lisa Ward and John Berlin. If you phone the office, it's most likely Lisa or John who will take your call. Lisa will be handling bookkeeping, filling orders, and being the PC contact person. John will be mainly concerned with class scheduling and serving as the Mac contact person.

If you're ever in Santa Barbara, they extend a warm welcome to you (1-805-963-1338).

-- * --

The meeting at Aston University (Birmingham, UK) is shaping up into a glorious international gathering of TeX users and luminaries: they'll be giving papers, acting as course instructors, providing workshops. Set on a modern campus, TUG '93 will be

an exciting place to be that last week of July (the 26th to the 30th). The booking form is now out, so get your copy and fill it in. And pay particular attention to the courses being offered this year, both before and after the meeting week: they're a combination of some of the regular courses associated with TUG meetings, along with some new ones.

The organising committee has been hard at work coming up with some interesting options (there's a trip to Stratford to attend a performance of Shakespeare's King Lear), so plan on making this a grand holiday.

Something else which is new this year: three conference rates. It used to be that there was one rate for TUG members, and one for everyone else. But we've been seeing an increasing number of other user groups whose members attend TUG meetings. To recognise the fact that, while they aren't members of TUG, they certainly are supportive members of a TeX user group, TUG meetings will now have a third rate, midway between those for TUG members and non-members. A small point, but one which has been somewhat irksome in the past. TUG, and the organising committee, extend a particular welcome to these people, and invite them to come and attend our meeting.

I hope to meet many of you at Aston this summer. Please come and say hello.

- ◊ Christina Thiele
5 Homestead Street
Nepean, Ontario
K2E 7N9 Canada
cthiele@ccs.carleton.ca

Editorial Comments

Barbara Beeton

T*EX* users are certainly aware that their “favorite text processing system” is used to typeset documents in many languages. There are some problems, of course — fonts, hyphenation patterns, etc., etc. Nonetheless, with determination and help from other users these can usually be overcome.

Recent issues of *TUGboat* have presented information on ancient Latin hyphenation and the Croatian Glagolitic alphabet, and in this issue is an article on the *pinyin* representation of Chinese. There is also an almost constant stream of queries on the electronic discussion lists concerning fonts or hyphenation patterns for various languages. For many of these languages there are no easily accessible, or affordable, typesetting tools other than *T**EX*.

Gaelic

While some of these languages are spoken by large numbers of people, some of them are spoken by only a dwindling few. I always find it encouraging to hear about successes with languages that not long ago seemed to be heading toward extinction.

One TUG member, captivated by what he saw and heard while he was in Cork a couple of years ago for the *T**EX90* conference, spent much of last summer in Ireland learning Irish Gaelic. Irish has been taught in primary and secondary schools for some time, but only relatively recently have there been schools where instruction is completely in Gaelic; of all the Gaelic languages, Irish is now one of the least endangered. There is a discussion list for *T**EXies* interested in this area: **ITALIC-L** (for “Irish *T**EX* And *L**A**T**E*X Interest Conference List”). Anyone interested in joining the discussion should send the message **Subscribe ITALIC-L your full name** to **listserv@irlearn.bitnet**.

The news has recently come to me that there may be a renaissance of Scottish Gaelic as well. This is even encouraged by the government through support of Gaelic television programming (news, a weekly soap opera, and an hour or two daily of other programs) and instruction in schools. There is a new college, “Sabhal Mor Ostaig” (“the big barn at Ostaig”), devoted to Scottish Gaelic and other topics central to Scottish traditional culture, such as piping and singing. Scottish Gaelic books are now set mainly in IPA (the International Phonetic Alphabet), but I understand that people involved in this area are watching the development of character set

standards to see how they might affect the ability to process the language effectively. I’d like to thank Phil Taylor for calling this to my attention.

Erratum: British hyphenation

It’s been pointed out (by Dominik Wujastyk; thank you very much) that a statement I made concerning British hyphenation (*TUGboat* 13, no. 4, p. 452), namely that “it’s considered bad style to hyphenate a word after only two letters” is somewhat extreme. As Dominik put it, “A glance at any of the hyphenation dictionaries for British English will immediately show hundreds of examples of *xx-* words.” The key, then, is to hyphenate according to etymology.

Anders Thulin, in a message to **comp.text.tex** (11 February 1993), presented some cogent comments on why syllabification is not necessarily an adequate guide:

It should probably be noted that most dictionaries show syllable division rather than hyphenation points. In many cases, they are the same, but many typographers, being more concerned with clarity than syllabification, won’t accept potentially misleading hyphenation points.

“anal-y-sis” is probably the most famous of these. anal-sis is the best hyphenation, since anal-ysis can lead the reader to assume some anal-related term is used.

[...] it would be only a minor confusion, and one that would be unlikely to remain. But if enough points of confusion are accumulated, the reader will probably stop reading altogether. The problem would not be with the text, but in the presentation of it: in the typography. And that kind of problem used to be avoided.

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

Philology

Typesetting Chinese *pinyin* Using Virtual Fonts

Wai Wong

Pinyin is a phonetic system for transcribing the pronunciation (in Mandarin) of Chinese characters. It is in widespread use in China [1]. *Pinyin* is taught to millions of children in schools there. It is also used to teach foreigners and speakers of other Chinese dialects to learn Mandarin. This paper describes a package for typesetting *pinyin* in L^AT_EX [2]. It uses the features of virtual fonts and ligatures to provide an easy way of generating fonts for *pinyin* and a simple input method. This package requires a .dvi driver which is capable of handling virtual fonts.

1 What is *pinyin*

Pinyin uses the Latin alphabet to transcribe the sound of Chinese characters. All Chinese characters are single syllables. Each syllable can be divided into two parts: the *initial* and the *final*. The initials are consonants (or shēng mǔ in Chinese). Some of the finals (or yùn mǔ in Chinese) are pure vowels, the others are combinations of vowels and consonants. Each syllable can be pronounced in one of four possible tones (shēng diào). When writing down *pinyin*, the tone of the characters is indicated by placing a *tone mark* on top of the main vowel in the finals. For example the vowel a can have the following four tones á á ǎ à.

There are two styles of writing *pinyin*: the first is to separate syllables by spaces, i.e., the *pinyin* for each character is separated, the second puts the *pinyin* of multi-character words together, like pīnyīn. When writing in the second form, syllables beginning with a, e or o may follow a vowel-ending syllable. This will cause confusion. If such case arises, an apostrophe mark (') is used to separate them. For example pí'aǒ (meaning leather jacket).

2 How to typeset *pinyin*

Since *pinyin* uses the Latin alphabet, it would not be too difficult to typeset it in L^AT_EX, except one has to place the tone marks properly. The normal L^AT_EX accent mark macros may be used for this purpose, but they are not very convenient since you need to put a tone mark on each syllable. (Try typing in

ā	a-	á	a'	ă	av	à	a'
ē	e-	é	e'	ĕ	ev	è	e'
í	i-	í	i'	ĭ	iv	ì	i'
ō	o-	ó	o'	ŏ	ov	ò	o'
ú	u-	ú	u'	ŭ	uv	ù	u'
ü	u:-	ü	u:’	ŭ	u:v	û	u:’
ü	u:	ê	ē				
Ā	A-	Á	A'	Ă	Av	À	A'
Ē	E-	É	E'	Ĕ	Ev	È	E'
Í	I-	Í	I'	Ĭ	IV	Ì	I'
Ó	O-	Ó	O'	Ŏ	OV	Ò	O'
Ú	U-	Ú	U'	Ŭ	UV	Ù	U'
Ü	U:-	Ü	U:’	Ŭ	U:v	Ù	U:’
Ü	U:	Ê	Ē				

Table 1: Input formats of *pinyin* characters

a paragraph of properly tone-marked *pinyin* using those macros yourself.)

The *pinyin* package provides a new style option *pinyin* to simplify the input and typesetting of *pinyin*. This option provides an environment, a simple input method and several fonts specially built for *pinyin*. To use it, put the word *pinyin* in the optional argument of the \documentstyle command. Then, within your document, the *pinyin* environment can be used anywhere.

Within the *pinyin* environment, tone marks are input by typing the appropriate character after the main vowel. The characters to mark the tones are: - ' v '. For example, ma- ma' mav ma' will produce mā má mǎ mà. Another example shown below is a poem by the famous poet lǐ bái in the Tang dynasty. The input

```
\begin{quote}%
\begin{pinyin}%
chuáng qíán míng yuè guāng\\
yí shì dì shàng shuāng\\
juv tó'u wa'ng mi'ng yue'\\
di- to'u si- gu- xia-ng
\end{pinyin}%
\end{quote}
```

produces

chuáng qíán míng yuè guāng
yí shì dì shàng shuāng
jū tóu wàng míng yuè
dī tóu sī gū xiāng

The input formats of all possible combinations of the tone marks and vowels are listed in Table 1.

	'0	'1	'2	'3	'4	'5	'6	'7	
'20x		Ā	Á	Ă	À	Ē	É	Ě	"8x
'21x	È	Í	Í	Í	Ì	Ó	Ó	Ó	"8x
'22x	Ò	Ū	Ú	Ü	Ù	Ū	Ü	Ü	"9x
'23x	Ü	Ü	Ê						"9x
'24x		ā	á	ă	à	ē	é	ě	"Ax
'25x	è	í	í	ý	ì	ó	ó	ó	"Ax
'26x	ò	ū	ú	ü	ù	ū	ü	ü	"Bx
'27x	ü	ü	ê						"Bx
	"8	"9	"A	"B	"C	"D	"E	"F	

Table 2: *pinyin* character positions

The *pinyin* environment can take an optional argument which specifies the font to use. The currently available fonts are **sf** and **rm**. Conventionally, *pinyin* is printed in a Sans Serif font, so **sf** is the default as you can see in the examples above. If you say

```
\begin{pinyin}[rm] ... \end{pinyin}
```

the *pinyin* text will be typeset in a Roman font.

Very often, a small piece of *pinyin* is embedded in running text. A shorthand command **\py** can be used instead of the *pinyin* environment. For example, **\py{pi-n yi-n}** produces pīn yīn. Like the environment, **\py** can also take an optional argument to specify the font.

There is a command to specify the size of the *pinyin* characters: **\pysize**. It takes a single argument which is the required size. It can be any reasonable size for text, such as 10pt, 11pt and so on. The default is 10pt.

3 The implementation of the *pinyin* package

The *pinyin* environment is implemented using virtual fonts and ligatures. Two families of virtual fonts, namely **pyrn** and **pyssn**, were created for typesetting *pinyin*. These *pinyin* fonts are needed because:

- accented characters, e.g., ā, ù, which are not included in ordinary TeX fonts and not even in the extended TeX (DC/EC) fonts, are required in *pinyin*;
- a special ligature table is required to support the input method in the *pinyin* environment.

These *pinyin* fonts are based on the Computer Modern fonts, i.e., **pyr10** is modified from **cmr10** and

pyss10 from **cmss10**. Ligature tables were defined for each of the vowels to map the combinations of the vowel and tone mark character to new character positions. All new characters and their codes are listed in Table 2¹. The compound characters are defined in terms of the base vowel characters and the appropriate accent characters. These mappings and character definitions were done in the property list files. For example, the **pyss10** font is created by the following procedures:

1. convert **cmss10.tfm** to **cmss10.pl** using **tftopl**;
2. copy **cmss10.pl** into a file named **pyss10.vpl**;
3. edit **pyss10.vpl** to add the ligature tables and new character definitions;
4. generate **pyss10.tfm** and **pyss10.vf** using **vptovf**.

The ligature tables for all *pinyin* fonts should be identical. Appendix A lists the complete ligature table. When editing the **.vpl** file, care must be taken to merge this into the existing table in the file.

The new character definitions have a general form that looks like the example below:

```
(CHARACTER 0 241
  (COMMENT a1 a-)
  (MAP (PUSH) (SETCHAR C a) (POP)
    (SETCHAR 0 26))
  (CHARWD R 0.480557)
  (CHARHT R 0.608887)
)
```

¹ The positions of the lowercase *pinyin* characters are arranged in a way that their codes are equal to the least significant byte of GB codes (the Chinese national standard). For example, ā at position "A1 has GB code A8A1 in hexadecimal.

for single accent in (̄, ́, ̂, ̃, ̄, ̅)
 if the base character *basechar* is in (a, e, o, u)
 wd is the width of *basechar*
 ht is the height of *accent*
 no movement is need for setting the character
 the program is:
 (PUSH) (SETCHAR *basechar*) (POP)
 (SETCHAR *accent*)
 if the base character *basechar* is i
 basechar becomes i (the dotless i)
 wd is the width of *basechar*
 ht is the height of *accent*
 the program is:
 (PUSH) (MOVELEFT (*wd_{accent}* - *wd_{basechar}*)/2)
 (SETCHAR *accent*) (POP)
 (SETCHAR dotless i)
 for double accent
 wd is the width of the base character (u)
 ht \leftarrow *ht_{ddot}* + *ht_{accent}* - *ht_u*
 the program is:
 (PUSH) (SETCHAR C u) (POP)
 (PUSH) (SETCHAR O 177) (POP)
 (MOVEUP *ht_{ddot}* - *ht_u*)
 (SETCHAR *accent*)

Figure 1: Algorithm for *pinyin* character definitions.

This is the definition for character ā whose character code is '241. The MAP property specifies how this character is generated. It first saves the current position, typesets the character a in the current base font and recovers the current position. It then overprints the accent character ̄ whose code is '26. This is the simplest case since the accents in CM fonts were designed so that they do not need to be raised or lowered for most lower case letters. For more complicated characters like ü, the second accent mark (the one on top) has to be moved up. The amount can be calculated using the heights of the base character and the accents given in the original .p1 file. The CHARWD and CHARHT properties specify the width and height of the new character. They are calculated from the widths and heights of the base and accent characters.

The algorithm in Figure 1 was used in building the *pinyin* fonts where *wd* and *ht* are respectively the width and height of the new characters. This gives a precise specification of the new characters. The resulting appearance of the compound characters is very good though not perfect. The advantage of using an algorithm is that it could be possible to

mechanize the generation of new font files by a program. To achieve the perfect result, manual editing, i.e., moving the accents or base character around, is necessary.

4 A sample installation

This package has been developed and tested in UNIX systems. It will not be difficult to port it to other systems; in fact, the style file and the font files can be moved to other systems without any change provided that the new environment has *TEX* and .dvi driver supporting virtual fonts. The remainder of this section describes an installation procedure in UNIX systems as an example.

This package is distributed as a uuencoded compressed tar archive file for UNIX users. Use the commands below to decode and extract the files:

```
uudecode pinyin.tar.Z.uue
zcat pinyin.tar.Z | tar xvf -
```

Afterward there will be a directory containing the style file *pinyin.sty* and the .vf and .tfm files. To install it, follow the procedures below:

1. copy *pinyin.sty* to the directory where *TEX* looks for macro files. In UNIX systems it is usually */usr/local/lib/tex/inputs*.
2. copy all the *.tfm* files to the directory where *TEX* looks for fonts. In UNIX systems it is usually */usr/local/lib/tex/fonts/tfm*.
3. copy all the *.vf* files to the directory where your *.dvi* driver looks for virtual fonts. For *dvips* in UNIX systems, it is usually */usr/local/lib/tex/fonts/vf*.

5 Conclusions and future work

The *pinyin* package provides a simple and flexible means of typesetting *pinyin* text. Its usefulness is obvious to anyone who wants to perform this task. It is very easy to use. The output quality is high, thanks to the *TEX* system.

Although the *pinyin* package currently has a small number of fonts and they are hard-wired in the *pinyin* environment currently, it is sufficient for most applications since *pinyin* is only a phonetic transcription, it is usually mixed with ordinary text in a document and seldom appears as an entire *pinyin* document on its own. Actually, having special fonts will provide a visual distinction between ordinary text and *pinyin*.

A natural extension to this work will be to automate the generation of new fonts by implementing

A Ligature table for *pinyin* characters

This appendix lists the ligature table for the *pinyin* characters. It should be merged into the existing table in the base font.

(LABEL C A)	(LIG C v O 217)	(LIG O 140 O 250)
(LIG O 55 O 201)	(LIG O 140 O 220)	(STOP)
(LIG O 47 O 202)	(STOP)	(LABEL C i)
(LIG C v O 203)	(LABEL C U)	(LIG O 55 O 251)
(LIG O 140 O 204)	(LIG O 55 O 221)	(LIG O 47 O 252)
(STOP)	(LIG O 47 O 222)	(LIG C v O 253)
(LABEL C E)	(LIG C v O 223)	(LIG O 140 O 254)
(LIG O 55 O 205)	(LIG O 140 O 224)	(STOP)
(LIG O 47 O 206)	(LIG O 072 O 231)	(LABEL C o)
(LIG C v O 207)	(STOP)	(LIG O 55 O 255)
(LIG O 140 O 210)	(LABEL O 231)	(LIG O 47 O 256)
(LIG O 136 O 232)	(LIG O 55 O 225)	(LIG C v O 257)
(STOP)	(LIG O 47 O 226)	(LIG O 140 O 260)
(LABEL C I)	(LIG C v O 227)	(STOP)
(LIG O 55 O 211)	(LIG O 140 O 230)	(LABEL C u)
(LIG O 47 O 212)	(STOP)	(LIG O 55 O 261)
(LIG C v O 213)	(LABEL C a)	(LIG O 47 O 262)
(LIG O 140 O 214)	(LIG O 55 O 241)	(LIG C v O 263)
(STOP)	(LIG O 47 O 242)	(LIG O 140 O 264)
(LABEL C O)	(LIG C v O 243)	(LIG O 072 O 271)
(LIG O 55 O 215)	(LIG O 140 O 244)	(STOP)
(LIG O 47 O 216)	(STOP)	(LABEL O 271)
	(LABEL C e)	(LIG O 55 O 265)
	(LIG O 55 O 245)	(LIG O 47 O 266)
	(LIG O 47 O 246)	(LIG C v O 267)
	(LIG C v O 247)	(LIG O 140 O 270)
		(STOP)

the algorithm in Figure 1 and incorporating the ligature table. This will provide us an easy means of creating new font when the need arises.

The way of implementing this package, namely by using virtual fonts and ligatures, can be generalized for other languages, such as Vietnamese, which have many accented and double accented characters. For such applications, the automatic font generation program is necessary, and style files which support more flexible font change and the new font selection scheme (NFSS) are also required.

References

- [1] Scheme for the chinese phonetic alphabet. Issued by the State Council of the People's Republic of China.
- [2] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, 1985.
- [3] Tomas Rokicki. *DVIPS: a TEX driver*.

◊ Wai Wong
Computer Laboratory
University of Cambridge
New Museum Site
Pembroke Street
Cambridge CB2 3QG
U.K.
ww@cl.cam.ac.uk

Hardware/Systems

A Multimedia Document System Based on TeX and DVI Documents

R. A. Vesilo and A. Dunn

1 Introduction

This paper examines the development of a multimedia document system based on TeX. Multimedia document systems involve the design of many complex components including editors, formatters, display systems and components to support the content types used (text, images, graphics etc.). By using TeX to do the formatting, using a standard text editor to enter the document text contents and define the document structure, and modifying a DVI previewer to include support for non-text contents, the amount of effort in developing a multimedia document system is greatly reduced.

Multimedia includes time independent information such as text, graphics and raster images and time dependent information such as audio and video. The added time dimension means that documents are structured not only in space but temporally.

Multimedia information is inserted into TeX documents by means of multimedia macros and the resulting document is called a multimedia TeX document. This document is formatted by TeX to produce an extension of the standard DVI file, called a multimedia DVI file, that contains embedded multimedia commands. Since it could not be foreseen all the different content types that potentially could be used, a flexible approach was taken by using standard terminology, based on a standard model for representing multimedia information, for defining the macros and embedded DVI commands.

In order to present multimedia DVI documents, an existing DVI preview program was modified to interface to the different I/O devices and to support the document constructs needed for multimedia. It uses available image and audio processing software; however, software to integrate the various components had to be developed.

The remainder of this paper describes an overview of the system structure, the structure and definition of multimedia TeX and DVI documents and a description of the presentation program.

2 System Structure

Multimedia systems are integrated systems, which can be defined as a group of programs and components that are logically part of the one system and work together to handle an application. This can involve a sharing of data, the movement of data from one component to another or the use of the same presentational format by the different components. Integrated systems often combine functions previously performed by a series of single-purpose programs, for example, word processing, spreadsheets, communications and graphics.

One of the main trade-offs in system integration is in using off-the-shelf components against using in-house developed components. In a highly integrated approach, which has a consistent user interface and gives the user the impression that they were dealing with an integrated multimedia document, system development can be a large task involving the development of large programs and makes it difficult to take advantage of off-the-shelf components. In order to reduce development effort, the system described in this paper uses a number of existing and off-the-shelf components.

The hardware structure of the system is built around an IBM PC/AT compatible computer under DOS and an Enhanced Graphics Adapter (EGA), which provides an inexpensive platform. It is equipped with:

- a mouse,
- an Ethernet interface,
- an audio interface using the Votan voice card with attached microphone and speaker,
- a Matrix PIP-512 frame-grabber card,
- a PAL-RGB converter,
- a 10 MHz RGB colour monitor (SONY PVM 1300E),
- a video camera (National WVP-A2N).

The system software structure has components grouped into different categories: editing/creation, formatting, presentation and translation/communications. Processing is also categorised according to information type, with text, image and audio processing by different editors, formatting programs and presentation program elements. The image editor also has associated with it a scaling program that is used to format images so that they fit into the space reserved for them. Translator programs convert between multimedia DVI documents and the ISO Office/Open Document Architecture (ODA) Office Document Interchange Format (ODA/ODIF) [ISO8613] but will not be discussed in this paper.

3 Multimedia TeX and DVI documents

3.1 Multimedia Contents Model

To deal with arbitrary content types a general multimedia information model was developed. The model is similar to the model used in the ODA standards. Multimedia information consists of the encoded multimedia data and multimedia attributes that are used to describe the structure of the multimedia data and control the presentation of the data. For example, in an image, the bit sequence describing the actual pixel values of the image would be the multimedia data and the attributes might include the number of pixels per line, number of lines and bits per pixel.

In order to describe different data encodings and attributes a two-level scheme is used. At the first level, multimedia information is divided into generic content types. These include images, audio and graphics. At the second level each content type is split into a number of different content architectures where each content architecture describes a particular method of data encoding and a particular set of multimedia attributes.

This two-level structure is implemented by having each piece of multimedia information described by a multimedia attribute file which contains the associated attributes and the name of the file containing the encoded multimedia data. The attribute files are text files, enabling them to be easily created and modified by text editors while the multimedia data files are usually binary files.

A further refinement is to have multimedia information defined as either internal or external. Internal multimedia information is presented on the same device as the body of the document while external multimedia information is presented on an auxiliary device, such as an external colour monitor or a loudspeaker. During document viewing, icons are used to indicate the presence of external contents and to activate the presentation of external contents.

A standard method for defining multimedia attribute file formats is used in order to provide consistency, flexibility and extensibility. Attributes appear on separate lines and are of the form:

name = value

Standard attribute names are used where possible, but the diversity of content architectures means there may have to be exceptions to this. Values are associated with a data type which include string,

integer, real, boolean and content architecture file extension. Comment lines begin with a ";".

3.2 Temporal Structuring and Scripts

With the added temporal dimension in multimedia documents, there is greater interaction with the user because of the need to control the playing of dynamic content types (in particular, audio in the present case).

Temporal structuring and user interactivity is achieved by means of icons and scripts.

In general, icons are used to indicate the presence of external multimedia information. They can be positioned either in the body of the document and be associated with related document contents or in a special section off the page, on another part of the previewer screen, and be associated with the entire page instead. The usual method of presenting external contents is to select the associated icon and then select the play function. Dynamic contents are a special case of external contents where the contents presentation is time limited. In these cases there is also the option of having the dynamic contents played automatically when moving to a new page, with the selection and playing operations conducted automatically. An example usage might be to have all the audio messages on a page played upon entry to the page.

Scripts are used to link a number of icons together and cause them to be played in sequence and can be used to contain simple audio, graphic and image presentations. Each icon involved has an identifier number and the sequence can be described directly by macro arguments or by a script file. The sequence of icons may be played one after the other with or without user intervention when going from one icon to the other. The individual icons in the script can also be played on their own.

3.3 Multimedia macros

The multimedia TeX macros used are shown in Figure 1.

Macros begin with the letters 'mm' to signify that they are multimedia. In the succeeding letters, 'img', 'geo' and 'aud' stand for images, geometric graphics and audio, respectively; 'int' and 'ext' stand for internal and external contents, respectively; and, 'on' and 'off' specify whether an icon representing external contents is to be appear on the document page or in the special icon area off the page. There are two variants of the script

\mmimgingt{id}{file}{arch}{hdim}{vdim}	insert internal image
\mmimgexton{id}{file}{arch}	insert external image, icon on-page
\mmimgextoff{id}{file}{arch}	insert external image, icon off-page
\mmaudioon{id}{file}{arch}{auto}	insert audio, icon on-page
\mmaudiooff{id}{file}{arch}{auto}	insert audio, icon off-page
\mmgeoint{id}{file}{arch}{hdim}{vdim}	insert internal graphic
\mmgeoexton{id}{file}{arch}	insert external graphic, icon on-page
\mmgeoextoff{id}{file}{arch}	insert external graphic, icon off-page
\mmscriptlist{id}{list}	insert a script list
\mmscriptfile{id}{file}	insert a script file

Figure 1: Multimedia macros

macro. One using a 'list' argument and one using a 'file' argument.

The icon identifier number argument 'id' is used to identify icons associated with a script. For those icons not associated with a script the id has to be set to zero.

The 'file' argument specifies the name of the multimedia attribute file. The 'arch' argument specifies the particular multimedia content architecture used.

Macros which insert internal contents (images or graphics) contain arguments specifying the horizontal and vertical dimensions, 'hdim' and 'vdim', of the space to be reserved in the document body to hold the contents.

Macros which insert icons into the document page require space to hold icons but this is reserved within the macro definition. Macros which position icons off the page reserve no space in the document.

The 'auto' argument, which is either 'Y' or 'N', specifies whether or not dynamic contents is to be played automatically upon entering a page.

The 'list' argument consists of identifier number separated by commas.

3.4 Multimedia DVI commands

Multimedia DVI commands are the mechanism by which information about multimedia information in a document is passed to the presentation program and are embedded into DVI files by the multimedia macros as string arguments of the *XXX* command (see [KNU86] for a description of DVI commands). They have the following format:

- The first two characters are 'MM' to distinguish the multimedia DVI command from other possible uses of the DVI *XXX* command.
- The next two characters are a two digit identifier for the particular command. There is a mul-

timedia DVI command corresponding to each multimedia T_EX macro.

- This is followed by a list of command parameters separate by blanks.

Each parameter consists of a letter, followed by '=', followed by the command value. The parameters used are shown in Figure 2.

The D, F, A, H, V and U parameters are user supplied while the rest are provided by the macro body.

3.5 Detailed multimedia macros

Details of representative macros are given below. The other macros are similar with minor changes.

The definition of an \mmimgingt macro, which inserts an internal image into the document, is as follows:

```
\catcode`\@=11
\def@mmimi#1#2#3#4#5{%
  \special{MM01 D=#1 F=#2 A=#3 H=#4 V=#5}}
\def\mmimgingt#1#2#3#4#5{\null
  \dimen0=#4\dimen1=#5\null
  \edef\hf{\the\dimen0}\null
  \edef\vf{\the\dimen1}\null
  \hbox to \dimen0
    {\vrule
      height\dimen1 widthOpt\null
      @mmimi{#1}{#2}{#3}{\h}\{v}\null
      \hfil}}
\catcode`\@=12
```

The \catcode`\@=11 command is used to allow private macro names to be defined for convenience. It is cancelled by the \catcode`\@=12 command at the end of the macro definitions. The #1 argument of \mmimgingt is the identifier argument. The #2 argument is the name of the attribute file. The #3 argument is the multimedia content architecture. Space for the image is reserved by means of a \hbox with image dimensions passed by the #4 and #5 macro arguments: #4 is put into register \dimen0 and made

Identifier	Description	Format
D	Multimedia content identifier	Integer
F	Attribute file name	Valid file name
A	Content architecture	Three character string
H	Horizontal size	Valid TeX dimension
V	Vertical size	Valid TeX dimension
X	Horizontal size of icon	Valid TeX dimension
Y	Vertical size of icon	Valid TeX dimension
I	Icon to be inserted	Single character (Y or N)
U	Automatic content presentation	Single character (Y or N)
O	Icon on the page	Single character (Y or N)

Figure 2: Command parameters used in DVI commands

the width of the `\hbox`; and, #5 is put into register `\dimen1` and made the height of a `\vbox` of zero width to define the height of the `\hbox`. This is done to position the `\special` command at the reference point of the `\hbox`. The `\@mimimi` macro is a private macro to invoke a `\special` command which inserts an `XXX` command, containing the multimedia DVI command, into the DVI file.

In the `\mmimgexton` macro, the `\hbox` is used to reserve space for the icon rather than for the internal contents, with the X and Y multimedia arguments being used instead of H and V. The I argument is set to 'Y' to indicate that an icon is to be inserted by the presentation program and the O argument is set to 'Y' to indicate that the icon is on the document page.

In the `\mmimgextoff` macro, no `\hbox` is inserted to reserve space. The I argument is still set to 'Y' but the O argument is set to 'N'.

4 Presentation Program

The multimedia document presentation program was implemented by extending an existing DVI preview program developed in-house at the University of Sydney, which was written in C and runs on an IBM PC compatible computer. The main functions of the existing previewer were to parse DVI documents, access fonts, display characters at the correct screen position, navigate through the document, scroll document pages and provide help facilities. Modules were added for it to have a mouse-based, graphical interface and hardware and software to support image display on the main screen, image display on an external monitor, audio presentation and user interactive control of audio and image presentation.

The extensions to the existing DVI previewer are such that the principles can be applied to a gen-

eral previewer, provided that the previewer has access to screen-interface facilities that support mouse movement, allow icons to be displayed and mouse clicks on icons to be detected.

The screen layout consists of four sections. The largest section, in the centre of the screen, is used to display the document contents. At the right hand edge are two narrow vertical columns, one containing mouse buttons to activate menu functions such as playing of external contents, navigating through the document and exiting the program, and the other column is used to display the icons declared as "off the page". At the bottom of the screen are a number of message boxes to display the current mode of the previewer, information about the duration of dynamic content types and error messages.

The system supports images in two in-house formats. Although other image formats are available, these two were readily available at the time and easy to incorporate into the system. The modular nature of the system means that other formats can also be added without too much difficulty. One image format is based on the FITS image format [WGH81], used for representing images in radio astronomy and can support colour. The other image format is for simple black and white raster graphic bit-map images. The Halo graphics library is used to display both image formats.

Audio is processed in two formats: unstructured and structured audio. Unstructured audio stores voice messages as binary files recorded using a stand-alone record program. The attributes in the multimedia attribute file are based on the voice algorithms used by the VOTAN voice card and include a flag to indicate compressed or uncompressed voice coding, the amount of compression, whether silent mode encoding is used, the duration of message in tenths of seconds and the length of voice file in bytes. In order to play an audio passage the audio icon is

selected by the mouse (if automatic playing is not enabled) and then the play menu function is activated from the right hand menu column. Playing can be interrupted through the keyboard.

Structured audio uses a three-level hierarchy based on passages, paragraphs and sentences created using a structured voice editor [Ng89]. The editor includes commands to insert, copy, paste, delete and record voice at either the passage, paragraph or sentence level. The voice editor is integrated into the presentation program to allow passages to be played under the control of the structured voice editor interface.

Graphics is supported in the form of Computer Graphic Metafiles (CGM) [ISO8632] but only to a limited degree due to the lack of an editor to create CGM metafiles.

A two-stage process is involved in displaying document pages. During the first stage, the DVI page is scanned, the multimedia DVI commands for the page are extracted, parsed and stored in a list and the text, icons and internal multimedia contents are displayed. Each internal (and external) multimedia content type and architecture has a separate module which is accessed by a standard interface. By separating multimedia dependent processing from the other types of processing, multimedia information can be handled generically, making the system easy to expand to include new content types and architectures.

During the second stage, mouse clicks are processed to display external multimedia information. This requires accessing the multimedia attribute and data files and calling the different device interface libraries. An escape mechanism is provided to save the current page display, invoke a new display and return to the previous display so that device specific user interfaces can be used.

5 Conclusion

This paper has examined the design of a multimedia document system based on TeX and DVI documents. The system used a number of off-the-shelf and existing components to reduce development effort. However, more attention needed to be paid to issues in integrating the system components as a result. The \special command was found to be an important technique for achieving this integration.

By applying standard methods for defining multimedia macros and multimedia DVI commands based on a general multimedia model, a flexible and extensible multimedia document was developed.

The system developed can easily be extended to include new content types and architectures.

The presence of dynamic contents in a document means there is more interaction between the user and the document. Two mechanisms for structuring dynamic contents were icons and scripts. By a combination of different multimedia DVI commands a range of interactions can be produced.

The effort in modifying the document preview program is reduced by using standard interfaces and modularity in the design. Once program changes are made, new features can be made available to users by copying existing macro definitions and making minor changes.

Further information regarding the multimedia document system can be obtained by contacting R. Vesilo at Macquarie University by electronic mail at rein@macadam.mpce.mq.edu.au.

Acknowledgments

The work conducted was part of a PhD thesis at the University of Sydney. The author would like to acknowledge his supervisors David Skellern and Robin King, the CSIRO Division of Information Technology, who helped fund the project and the Commonwealth Postgraduate Award Scheme.

References

- [ISO8613] ISO 8613, Information Processing Systems - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format. 1989.
- [ISO8632] ISO 8632, Information Processing Systems - Computer Graphics - Metafile for the Storage and Transfer of Picture Description Information. 1987.
- [Knu86] D. E. Knuth. TeX: The Program. Addison-Wesley, Reading Massachusetts, 1986.
- [Ng89] D. Ng. Multimedia Communications. Masters thesis. University of Sydney, 1989.
- [WGH81] D. C. Wells, E. W. Greisen, and R. H. Harten. FITS: A Flexible Image Transport System. Astron. Astrophys. Suppl. Ser., Vol. 44, 1981, pp. 363–370.
- ◊ R. A. Vesilo
School of Mathematics, Physics,
Computing and Electronics
Macquarie University
NSW, 2109, Australia
- ◊ A. Dunn
School of Electrical Engineering
University of Sydney
NSW, 2006, Australia

Book Reviews

Review of: Modern T_EX and Its Applications

Jon Radel

Michael Vulis, *Modern T_EX and Its Applications*. CRC Press, 1993. vii + 294 pp. + floppy disk ISBN 0-8493-4431-X.

This work is a departure from the current stream of books on T_EX in several ways, the most important being that it documents VT_EX, which the author himself refers to as a heresy, rather than T_EX — more about this later. The author also claims that this book is aimed at the uncovered middle ground between the guides for the utter beginners and the reference works for the T_EX professional, a tricky position to attain and not really as uncovered as Michael Vulis claims. And while I did find a total of three T_EX books published in the U.S. since 1984 which do not mention the T_EX Users Group on my shelf, this appears to be the only one that references TUGboat articles repeatedly while not mentioning TUG once.¹

1 For the Technically Oriented Beginner

The bulk of this book is a quietly unassuming, but very practical, guide to how to use Plain T_EX, with ample material on the extensions of VT_EX. I think the sequence of the book would be most appealing to a user with a background in microcomputer based wordprocessing software, no background in T_EX, little interest in the clever transfer of traditional typesetting technology to the idiom of T_EX's internals, and enough technical orientation to move quickly through this material. It tells the reader how to set margins and pick fonts long before it mentions glue, box variables or tokens, and the reader is supplied ample practical examples but very little information about T_EX's insides.

After the introductory chapter covers basics: \input, \bye, why one doesn't use a wordprocessor that puts control characters in the file for T_EX input, and enough other information to allow the user to T_EX his first file, the book moves on to a chapter on "Fonts and Characters". It is here that the differences between T_EX and VT_EX become

¹ Since insufficient bibliographic information is given on all other works referenced, this isn't really a snub — just sloppy.

very apparent. The primary extension in VT_EX is its support for scalable, vector fonts in both VT_EX itself and the matching drivers. This allows for such commands as \font\myfont=mvpair scaled 1200 aspect 600 bold 04 slant 500. After this \myfont would refer to a nice looking serif font made hideous by scaling to 12 points, squashing down to 60% of its original height, making slightly bolder, and slanting 22 degrees to the right. While this example I have concocted is rather useless, the point is made that given a couple of vector files, which are already smaller than the .PK files of standard T_EX, the user can create slanted characters, bold characters, small capitals, etc., on the fly without any additional disk space being used for font storage. This feature alone has won over at least one T_EX user with a laptop [1].

Unfortunately, in this chapter the author's biases get in the way of reporting the current state of affairs for users of T_EX, as opposed to VT_EX. For example, while Computer Modern Roman has become the face that many love to hate, the author's dismissal of it, with "Of the three [primary text fonts in the CM family], Roman is a fairly weak clone of the Century SchoolBook," took me by surprise coming after one of the better two page histories of typefaces that I've read. It is true, I suppose, that the target audience for this book isn't interested in the full history behind Donald Knuth's choice of Monotype Modern No. 8A, but the author's assertion is a rather abrupt one sentence history of the font.

One could get the impression from this chapter that all T_EXs other than VT_EX have been standing still for some time. When making his arguments about vector fonts saving significant disk space, he gives the outmoded .PXL files slightly more prominent billing than the more compact .PK files that all of VT_EX's competitors have been using for years. He also mentions the Almost Computer Modern (AM) alongside the Computer Modern (CM) fonts without any explanation of the distinction or of the fact that AM fonts were replaced by the CM fonts over half a decade ago. Virtual fonts, not too surprisingly, get no coverage at all.

Luckily, things pick up again as the book covers topics where VT_EX differs little from T_EX. Chapter 3 covers "Formatting", with coverage of headers and footers, skips, how paragraphs and pages are set, footnotes, and how to deal with the various problems surrounding these. In keeping with the goal of the book, there is plenty of practical advice on how to deal with issues such as under and overfull boxes, without an excess of technical detail. The same is true of Chapter 4, which covers "Math

Mode" with a large number of examples and a few details, such as why one should not reassign fonts to math font families 2 and 3 without great care.

By Chapter 5, "Variables, Dimensions, Glue", the coverage necessarily gets somewhat more complex and technically oriented. It, Chapter 6, "Macros", Chapter 7, "Input/Output and Extensions", Chapter 8, "Modes, Rules, Boxes", and Chapter 9, "Tabulation and Tables", give a through coverage of everything needed to do basic macro writing and layout tasks under Plain *T_EX*. There is, however, insufficient information for the user to write his own complex formats. For example, `\shipout` and `\output` are covered in about a page. The reader is left with the knowledge of roughly what they do, that `\output` is what you revise if you want double-column output, but no knowledge at all of how to go about such revisions.

The last chapter, "Font Rotation", is a bit of an oddball. It starts with an abridged version of Alan Hoenig's *TUGboat* article on the topic [2], followed by an explanation of how this is better done under V_{T_EX}. I was left a little confused as to the point; while this is terribly clever and interesting, I'm not sure how many people will find it of use. Personally, I would have found an expansion of the section on Cyrillic fonts, to one with general coverage of setting non-English texts, to be of greater practical use than 10 pages on how to typeset on circles. On the other hand, those who are switching from a wordprocessor to *T_EX* might indeed be interested in such effects.

The penultimate chapter is an appendix, with the font tables requisite to any *T_EX* manual, though this one features the V_{T_EX} fonts prominently, a bit of information on graphics inclusion, most of the macros which are supplied with V_{T_EX} as an extension to Plain, and a two page guide to the enclosed demo version of V_{T_EX}. The book closes with chapter 12, "Index". Unfortunately, it's incomplete, missing completely such commands as `\aliasfont`, one of the more important additions in V_{T_EX}.

2 Problems

There are two problems that keep me from recommending *Modern T_EX* freely as a good working manual for technically oriented *T_EX* users who have no intention of writing their own complex suites of macros, or of using L_AT_EX or other existing packages that go beyond Plain. The first is that the coverage of V_{T_EX} specific features is firmly entangled with that of features available in *T_EX* also. Not only are the fonts more flexible under V_{T_EX}, but there are a variety of primitives, such as `\random`, which re-

turns a random number, `\sincos`, which calculates a sine and cosine of an angle, and macros that extend those in *plain.tex*—though these the user can find in the appendix or on the included disk. While these extensions are not necessary, as most *T_EX* users prove on a regular basis, it would be frustrating to work with a manual containing intriguing examples you couldn't use. Naturally, this isn't a problem if you use V_{T_EX}. Given that, the demo software, and the \$100 off coupon for a V_{T_EX} system included in my copy, I'm sure that this book will serve as persuasive marketing for V_{T_EX}.

The second problem, unfortunately, affects all readers. While I found little that I could point at and say, "That is completely wrong" (but see below), the book is riddled with a variety of inconsistencies and imprecisions. For example, Computer Modern Typewriter is referred to in various places as "TeleType", "teletype" and "typewriter" without an explanation or pattern that I can discern. While that one is unlikely to confuse the average user, others, such as the statement on page 31 that the name of a font "may only contain characters and not digits or other non-characters" stand a good chance of confusing any reader who doesn't quickly realize that "letters" is meant in place of "characters" here, while the terms are used in the usual fashion elsewhere in the book.

There are also some explanations that could be more precise without necessarily being made more complex. For example, page 12 has an explanation of the escape character that includes the following text: "Built-in command names (as opposed to user-defined ones) must be blocks of letters without embedded punctuation marks or spaces (`\centerline`, not `\center line`, for example), or they must consist solely of non-letter symbols (`\!` or `\$`, for example)." While wonderful things can be done if you start changing category codes, under normal circumstances I am doubtful about leaving the beginning user with the impression that he can embed spaces in control sequences he defines, or use multiple non-code 11 (non-letter) characters. Removing the distinction between "built-in" and "user-defined" commands, and appropriate application of the word "single" would give something I consider of greater utility to the beginner. On the other hand, maybe the author is simply confused on the distinctions between control symbols and control words, as on the next page he states, "spaces after a command are ignored." This is followed by an example: "Similarly, '`\$5`' is equivalent to '`\$5`'. You should type '`\$5`' to obtain [a dollar sign and '5' separated

by a space]." Even in VTeX the first part of this example is not true.

It is also unfortunate that the generally well thought out examples—they are both amusing without being overbearingly cute and generally economical in the number of TeX features they showcase per example—are marred by errors. Some are silly and more likely to amuse than confuse (did you know that if you supply VTeX with "A Quick Brown Fox Jumps", it will typeset "The Quick Brown Fox Jumps"?). Others will cause the new user, already struggling with the concept of ligatures and the fact that real typesetters think opening and closing quotes are different, no end of confusion. The monospaced text, which purports to be a representation of the user's input, reduces various multi-character input sequences to a single monospaced character. On page 45, after explaining how to input hyphens, dashes, and quotes, an example is given that uses all of them. Until I looked at the output, I couldn't tell which was supposed to be an en-dash and which an em-dash, though the hyphen is a bit shorter. Any reinforcement of the notion of typing one, two, or three hyphens is lost.

Another class of comments by the author that I find a bit troubling is where he sells non-VTeX TeX short by making dubious claims about what it can't do. They range from his comment that the lack of a dollar sign is "a layout bug in some of the Computer Modern fonts," something I would not consider a bug but rather a design decision that was, in hindsight now that we have 256 character fonts, less than perfect (my computer science background may leave me too sensitive to the distinction between design shortcomings and implementation bugs) to far more serious ones. After discussing how to get upright quotes, German low quotes, and guillemets in VTeX, stating "Under standard TeX, none of these characters is available" would, in my opinion, leave the reader with the impression not, as is the case, that such characters are non-trivial to make use of in Plain TeX, but that if you aren't using VTeX, you can't use them at all.

Of course, a very close reading will uncover the ironic combination of several of the above tendencies: After insinuating that a degree symbol is available only in VTeX, by listing it on page 47, in a section headed "The following additional symbols are supported by VTeX," an example on page 89 contains \$91^\circ\$ rather than the VTeX equivalent, 91\circ.

3 The Demonstration Software

The book comes with a single floppy ($5\frac{1}{4}$ " high density, though $3\frac{1}{2}$ " is available on request) containing a demonstration version of VTeX. A few of the pieces that come with the full program are missing, such as the integrated editor, FAX output support, and some of the fonts. Upon use you quickly notice a 72 point "DEMO" in the middle of every page. Despite this, the software includes ample capability to experiment with VTeX, and includes printer support of LaserJet, DeskJet, PostScript, Bubble-Jet and Epson dot matrix printers. Quite frankly, if it weren't for that "DEMO", I would have written today's batch of outgoing letters using VTeX, as some of the supplied fonts are a nice change from Computer Modern.

Unfortunately, it is not possible to tell from the supplied information whether some of the problems I had (such as installing L^AT_EX under VTeX) were artifacts of the demo software, or are also present under the full VTeX. Presumably, there exists a utility for adding new .TFM files to the library of such files that VTeX uses, as I could not get the demo version to recognize the individual .TFM files for the L^AT_EX fonts.

4 The Heresy of VTeX

It is with some trepidation that I bring this up at all, as some of the discussion on this topic has gotten heated, for example during the question period after Michael Vulis presented his paper at the 1990 TUG conference [3]. On the other hand, he is not shy about it in this book:

The second heresy was the support of modern scalable font technology in VTeX, the MicroPress' implementation of TeX. VTeX enhances TeX in several ways; being the leading deviant dialect, it receives very substantial attention in this volume, since if TeX is going anywhere, it is all but certain that this is the direction it will take.

No mention is made of another "heresy" of growing popularity, TeX-XET, which allows for bi-directional typesetting. He gives further thoughts on the matter of keeping TeX alive by extending it in a later paper [4], and the material in both of these papers is presented in *Modern TeX*. In some ways the extensions of VTeX are quite superficial: scalable font technology and a few handy primitives that neither are vital nor impact the fundamental

workings of \TeX . In contrast, some of the underlying assumptions and limitations of \TeX are being challenged in papers, such as that presented by Philip Taylor at Euro- \TeX '92 [5], and discussions on the NTS-L e-mail discussion list. True, V \TeX is a shipping product, and the New Typesetting System (NTS) is not even close to that stage, but the vision of \TeX 's future offered by V \TeX is a limited one.

References

- [1] A.G.W. Cameron, "The Airplane Workstation." *Personal Workstation* 2(6):14–17 (June 1990).
- [2] Alan Hoenig, "Circular Reasoning: Typesetting on a Circle, and Related Issues." *TUGboat* 11(2):183–190 (June 1990).
- [3] Michael Vulis, "V \TeX Enhancements to the \TeX Language." *TUGboat* 11(3):429–434 (September 1990).
- [4] Michael Vulis, "Should \TeX be Extended?" *TUGboat* 12(3):442–447 (September 1991).
- [5] Philip Taylor, "The Future of \TeX ." *TUGboat* 13(4):433–442 (December 1992).

◊ Jon Radel
 P.O. Box 2276
 Reston, VA 22090-0276
jon@radel.com

Review of: Mathematical \TeX by Example

Philip Taylor

Arvind Borde, *Mathematical \TeX by Example*. Academic Press, 1993; \$19.95 (U.S.); ISBN 0-12-117645-2; 356 pp; not quite as deep as Crown 4to.

In *Mathematical \TeX by Example*, Arvind Borde continues in the mould which he pioneered in his earlier work *\TeX by Example*¹ — a style which eschews the discursive didactic approach and, with the absolute minimum of introduction, brings the reader face-to-face with real \TeX code, presenting on the facing page the typeset material which results from the use of this code. The code is not simply presented as a *fait accompli*, but is accompanied by copious notes (set as footnotes) which explain the significance of each of the \TeX commands used, and which give a general commentary on the purpose and implications of the code. The advantages of this approach are obvious: the reader is not bemused by meandering discussions about topics which are as yet merely vague concepts, but is instead presented with real-life instances of \TeX code and, at the same time, shewn the typographic effects which this code accomplishes. Arvind prepares the reader for the eponymous Examples with a mere 13 pages of introduction (14, if the preface is included) and then plunges into his subject with vigour, presenting as a first instance of his method the code and text which generate the facing page, which itself forms the introduction to Example 1. The code used in this first instance is both simple and powerful: much is achieved simply by \inputting two files, and more is achieved by assignments to (what are presumably) token-list registers and count registers. Sadly, at this point, it becomes only too clear that the material has not been proof-read by a \TeX expert: two glaring errors leap out from the very first page of code, and the aware reader is led to believe that the book *may* not turn out to be all that it is cracked up to be. The errors which so readily manifest themselves are not critical — one, indeed, is simply that the comment on line 2 is apparently an echo of that on line 1, rather than being the correct comment for the second line, whilst the other occurs in the footnotes, where \EndPage at line 26 is said to be '*an abbreviation for \hfil\break*' (one suspects that it is really an abbreviation for \vfil\reject). But

¹ *\TeX by Example*: Borde, Arvind; 1992. Published by Academic Press at £13-00 (U.K.), \$19-95 (U.S.); ISBN 0-12-117650-9; 178 pp; not quite as deep as Demi 4to.

another, more serious, infelicity dogs this very first page—lack of consistency, in the use of three different variants of the assignment statement: with a `<becomes>` operator, with a `<space>`, and with neither (lines 5, 30 & 32 respectively). This I regard as a serious error in any pedagogic work, but particularly in one which seeks to teach by example; for not only is the reader led to believe that no one form is preferable to the others, but, more seriously, doubt is raised in the percipient reader's mind as to exactly *why* the different forms are being used and what subtle distinction is being made (only much later, when the reader is already TeXnically literate, will he or she come to realise that these different forms merely represent an inexcusable example of what Fowler would term '*elegant variation*').

On turning the page, we are presented with a fairly daunting demonstration of mathematical TeX and, on the facing (*recto*) page, the resulting (and somewhat less daunting) page of mathematics. To be fair, the mathematical TeX is, upon closer inspection, not particularly advanced, and indeed probably represents about as gentle an introduction to mathematical TeX as might be reasonably expected in a work of this nature; we are introduced to in-line and displayed mathematics, subscripts and superscripts, the prime as implicit superscript, equation numbers, mathematical Greek, square roots and a bordered Hessian matrix. No attempt is made to factor out common elements at this stage, and the markup adopted is strictly functional. Again there are subtle inconsistencies: for example, a footnote refers to '*vol. 1 28, pp. 68--96*', with a control space after `vol.` but a normal space after `pp.`; thus `vol.` takes its correct, normal, space, whilst `pp.` takes a totally unjustifiable end-of-sentence space. Finally on this page, we note that in the footnote, '*Crell's Journal*' is letter-spaced by hand, with an explicit `\thinspace` (abbreviated to `\:`) between each pair of letters; this is consistent with the functional markup being used at this stage, but nowhere, so far as can be determined from the index, does the author develop a more general `\letterspace` command.

The next page introduces `\displaylines`, yet, remarkably, neither is it used to produce multi-line displays nor is its use justified in the footnotes in any way; the reason for its use at this stage remains somewhat obscure.

By the fourth page of code, macro definitions are introduced (by example, of course); one to place a bar at a constant height over its parameter, one to use this bar in a frequently recurring expression, and a third to produce what might best be described

as a bird's foot, by overstriking a `<less-than>` and a `<minus>` (doubtless mathematicians will rush to tell me the correct name for this symbol!). Here, too, the first hint of sophistication appears, with the bird's foot being defined as a `\mathrel` operator, to automatically achieve the most appropriate spacing.

Thereafter, each new page introduces new (and usually more sophisticated) techniques, whilst the markup moves from being purely functional to increasingly descriptive, high-level and content-oriented (which, interestingly, also models the development of the typical (better) TeX practitioner). I suspect that the mathematics becomes increasingly complex to match the increase in sophistication of the markup, but as a non-mathematician I am not in a position to judge the truth of this assertion.

Example 7 marks a significant watershed in the development of the book: whilst all previous examples have been in set in Computer Modern, Example 7 introduces for the first time the possibility of typesetting in other fonts, being set in Computer Concrete with mathematics in *AMS* Euler. Example 8 continues this trend by being set in Times Roman and MathTime, whilst Example 9 is set in Lucida Bright and LucidaNewMath. Example 10 continues the exploration of new territory by demonstrating the possibility of in-line POSTSCRIPT graphics and the inclusion of POSTSCRIPT files. Examples 11 and 12 demonstrate an alternative approach to graphics, using Michael Wichura's PJCTEX. Example 13 demonstrates the use of the LATEX picture environment within the medium of Plain TeX, and also gives samples of Piet Tuteelaers' chess font. This concludes the diversion into graphics.

Example 14 marks the second watershed in the book, by considering for the first time formats other than Plain TeX, and in particular considers *AMS*-TeX. Examples 17 and 18 demonstrate some of the possibilities of *LAMS*-TeX, and conclude Chapter 2 of the book (Chapter 1 was the Introduction; Chapter 2 is composed solely of the Examples from which the book takes its name).

Chapter 3 summarises the features of *AMS*-TeX, and compares and contrasts these with those of Plain TeX; Chapter 4 performs a similar function (although in less detail) for *LAMS*-TeX and *AMS*-LATEX. Chapter 5 is devoted solely to a discussion of fonts, including Computer Modern, POSTSCRIPT, American Mathematical Society (Euler and maths), Computer Concrete, LATEX symbol, and the 256-character DC/EC fonts, as well as the little-exploited but very powerful virtual font mechanism.

Chapter 6 returns to the main theme of the book by documenting, in detail, the *TEXniques* which were used to typeset the book; in particular, the method by which the ‘page-within-a-page’ format was implemented, and all the ‘assumed’ macros which were used to simplify and rationalise the markup. Here may be found, for example, the true definition of `\EndPage` (which turns out to be `\vfil\eject` as suggested above, rather than `\hfil\break` as claimed within the text); here too may be resolved any other contradictions which may have originated from an occasionally careless footnote. The code is excellently laid out using natural indentation wherever possible, and frequently (although not invariably) with vertical alignment of `\if`, `\else` and `\fi` (a technique which many other authors would do well to consider). If I were to offer any criticism of the layout of the code, it would be to suggest that the ‘left-hand side’ of a macro definition (`\def`, the control sequence or active character, and the parameter list) is worthy of a line in its own right whenever the replacement text of the macro runs to two or more lines; to concatenate it with the first line of the replacement text reduces the legibility of both. I might also suggest that open and close braces spanning several lines of code are worthy of the same high standards of alignment which are afforded to the `\if`, `\else`, `\fi` referred to above: it is never easy to find closing braces when they are both horizontally and vertically displaced from the open braces which they match. Considering the code itself, as opposed to the manner in which it is presented, it is what might be termed true plain *TEX*: refreshingly free of commercial-ats (except where they are essential), no obvious obsession with `\expandafter` and its friends, and a level of sophistication which should not be beyond the ambit of any potential reader—familiarity with *TEX* up to the level of `\mark` and its cohorts is all that is required. The same criticism of inconsistency of representation as was made of the main text is equally applicable here, however, and amongst the more perplexing features are the author’s occasional tendency to leave one space between a control word and a following close brace; under normal circumstances this is perfectly OK, but one wonders why the author felt it necessary—only in code considerably more sophisticated than that presented here might such a space become truly essential. More seriously, the frequent use of two or more instances of the same explicit numeric, dimension or glue literal would suggest to the less-aware student of this book that it is safe to ignore what is generally regarded as a fundamental precept of good programming style: if a constant is to be used more

than once, it warrants being given a *name*, both so that the reader can be certain that it is the *same* constant being referred to, and to protect the author from accidental changes to one instance of the constant not being reflected in all others.

The penultimate section of the book is a comprehensive bibliography, in which one is not surprised to find Knuth’s entries composing 42% of the total; the majority of the other authors cited are recognised authorities in their own fields.

The final section is simply entitled ‘*Glossary/Index*’, and one might be forgiven for initially affording it little interest. And yet, in the opinion of this reviewer, this chapter represents one of the greatest strengths of the whole book, for in its modestly-titled pages all relevant *TEX* primitives, Plain *TEX* and *AMS-TEX* commands, as well as those developed specifically for this book, are discussed and/or defined. Knuth’s asterisk convention for denoting *TEX* primitives is followed, and *AMS-TEX* commands are tagged with a small *AMS*. In essence, it is a mini *TEX* dictionary/encyclopaedia, and contains a wealth of invaluable information (with the occasional error) in its 110 pages.

A comparison of *Mathematical TEX by Example* with its earlier stablemate *TEX by Example* (of which I received a complimentary copy by courtesy of the author) yields some interesting comparisons: Arvind has clearly learned by experience, and in his second work avoids many of the slightly amateurish touches which marred *TEX by Example*. Gone, for instance, is the double spacing which created such a bad impression in much of the earlier work, and the overall trim size is reduced from (slightly less than) Demi 4to. to (slightly less than) Crown 4to., leading to a much more manageable volume. The dropped caps, too, presented in a quasi-illuminated manner, add a pleasing touch of class.

In assessing the overall merit of a book such as this, one must consider various parameters: the intended audience, the value to such an audience, the accuracy and precision of the text, the relevance of the examples, and to a lesser extent, the degree to which the book overlaps others in its field and outshines (or is outshone by) them. For *Mathematical TEX by Example*, the intended audience is clear: practicing mathematicians and mathematical secretarial staff who are, in the words of the author, ‘already broadly familiar with the basic uses of *TEX* [and who seek an introduction to] additional tools and techniques — mainly related to typesetting mathematics’. There is little doubt that such an audience will derive great benefit from this book — it covers a wide range of mathematical typesetting problems,

and gives not just a description of the TeXniques which might be used to solve them, but instances of the actual code; in that sense it is, therefore, a recipe book, and accordingly it possesses both the strengths and the weaknesses which typify books of that *genre*. If the problem requiring a solution is covered in the book (i.e. if its recipe is given), then the problem is completely solved; but if the problem is *not* covered (i.e. if no such recipe can be found), then the reader is left little wiser. Fortunately, by virtue of the very comprehensive Glossary/Index with which this book concludes, this deficiency of the recipe-book approach is mitigated, for even if the exact problem is nowhere discussed within the Examples, there is an excellent chance that the elements of the problem are at least alluded to in the Glossary/Index. Yet this approach, too, has its limitations: for example, the text on pages 16/17 reads '*may be written with the aid of a bordered Hessian, as follows:*', but if one turns to the Glossary/Index, knowing full well that one needs to typeset a '*bordered Hessian*', no such entry exists. One is led round the houses, via \bordermatrix (which yields a dead-end, at least in the context of this enquiry), via **matrices** (Plain TeX), to \$\matrix\$ (Plain) which finally yields the relevant page number (amongst many others). The Glossary/Index could therefore be improved (in the opinion of this reviewer) by containing key phrases which identify specific instances amongst the examples. In terms of accuracy and precision, the text is less than perfect, but is not so severely flawed as to render it useless; I recommend to Arvind that he employ a TeXnically literate proof-reader for any future volume, for whilst his approach is excellent, the inconsistencies and occasional real errors which populate this book do tend to detract from its value. Finally, in terms of the competition, *Mathematical TeX by Example* has one great advantage: it (and its stablemate, *TeX by Example*) are unique in their approach (at least, as far as I am aware; I have encountered no other TeX books which seek to educate solely through example); for those, then, who prefer to learn by osmosis rather than through orismology, this book is to be recommended, although perfectionists might do well to wait for a second edition, in which one hopes the errors and inconsistencies will have been eliminated.

◊ Philip Taylor
 The Computer Centre
 RHBNC
 University of London
 Egham, Surrey TW20 0EX, U.K.
 P.Taylor@Vax.Rhbnc.Ac.UK

Macros

The Bag of Tricks

Victor Eijkhout

A 144 point Hello! to you all.

On **comp.text.tex** I see with some regularity questions about actions on a whole page. For instance, how to put a box around a page, or how to overlay a piece of text on each page. Here is a solution based on redefinition of \shipout. Doing things this way has the advantage that it does not depend on the format used.

If you have more than one application like this, you could for instance put the following macros in a file **wholepage.tex**:

```
\let\xshipout\shipout
\def\shipout{\futurelet\SomeBox\yshipout}
\def\yshipout
{ \ifx\SomeBox\box
   \let\next\ShipZero
 \else \ifx\SomeBox\copy
   \let\next\ShipZero
 \else
   \let\next\ShipAfterZero
 \fi\fi
 \afterassignment\next\setbox0=
 }
\def\ShipAfterZero{\aftergroup\ShipZero}
```

This saves the old \shipout, and defines a new one that first investigates what kind of box is coming up. This box is then put in \box0, and a call to \ShipZero processes this and really ships it out by a call \xshipout\box0.

For every specific application a definition of \ShipZero has to be made. Here is a way to do overlays:

```
\newbox\OverlayBox
\def\ShipZero
{ \setbox0\vtop{\kern0pt \box0}
 \setbox0\vtop{\kern0pt
   \vtop to \Opt{\kern0pt
     \copy\OverlayBox\vss}
   \nointerlineskip \box0}
 \xshipout\box0 }
```

If you put these macros into a file **overlay.tex** followed by a line \input wholepage, then you can make an overlay by, for instance

```
\input overlay
\setbox\OverlayBox
\hbox to \hsize{\hfil\tt TEST}
```

Text ...

(If you use L^AT_EX, you create a file *overlay.sty* that can be specified on the \documentstyle line.

Cropmarks are a bit harder. I arrived at the following after a lot of tinkering. There are dimensions for the length and width of the crop lines, and for how far they are separated from the page.

```
\newdimen\croplength \croplength=20pt
\newdimen\cropsep \cropsep=10pt
\newdimen\cropwidth \cropwidth=2pt
```

Sometimes you may want to add some extra padding on the top, bottom, or sides.

```
\newdimen\croppadtop \croppadtop=0pt
\newdimen\croppadbot \croppadbot=0pt
\newdimen\croppadlr \croppadlr=0pt
```

Here are the crop lines.

```
\def\crophrule{\vrule
  height\cropwidth depth0pt
  width\croplength}
\def\cropvrule{\vrule
  width\cropwidth depth0pt
  height\croplength}
```

The following macro does the real work: it takes \box0 and puts the crop lines around it. You see that there are a lot of kerns of size .5\cropwidth: these make sure that cropping through the middle of the lines will give you exactly the page.

```
\def\ShipZero
{\setbox0\vbox
 {\offinterlineskip
  \dimen0\cropsep
  \advance\dimen0\croplength
  \setbox2\hbox to \wd0
   {(\kern-.5\cropwidth\kern-.5\cropwidth
    \cropvrule\hfil\cropvrule
    \kern-.5\cropwidth\kern-.5\cropwidth)
  \setbox4\hbox to \wd0
   {\llap{\crophrule
     \kern\cropsep\kern\cropwidth\%
     \hfil
     \rlap{\kern\cropwidth\kern\cropsep
       \crophrule}}
  \kern-.5\dimen0 \copy2
  \kern\cropsep\kern-.5\cropwidth
  \copy4
  \kern-.5\cropwidth
  \kern\croppadtop
  \box0
  \kern\croppadbot
  \kern-.5\cropwidth
  \box4
  \kern-.5\cropwidth\kern\cropsep}}
```

```
\box2
}\xshipout\box0 }
```

Finally, here's how to put a box around the whole page:

```
\newdimen\padding \padding=3pt
\def\ShipZero
{\setbox0\hbox
 {\vrule\kern\padding
 \vbox{\hrule\kern\padding
 \box0
 \kern\padding\hrule\%
 \kern\padding\vrule}
 \xshipout\box0 }}
```

And that's all for now.

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

[Editor's note: The edges of the page are assumed to be at the edges of the printed area, excluding any material that is "lapped" outward. (In plain T_EX, this "lapped" area includes the \headline, if any.) Readers who are preparing camera-ready copy will want to add sufficient padding to accommodate necessary margins. For two-sided presentation, they may also find that the gap between edge of page and crop mark should be different on the left and right. These macros can be adapted for that situation by replacing \croppadlr by \croppadin and \croppadout to indicate inside and outside margins, and specifying these appropriately on odd or even pages.]

Anchored Figures at Either Margin

Daniel Comenetz

1 Setting Figures

A figure in a box can be placed in text at one margin or the other, by measuring the box and adjusting the paragraph shape parameters so as to allow room for it. Macros that try to accomplish this automatically must be resourceful enough to decide what to do in a variety of special circumstances; the correctness or appropriateness of each decision depends on the requirements of the user. In my case, I need to set figures in mathematics text, and I have to be concerned with three problems, namely: (1) what happens if there isn't enough room left on the page for the figure at the point in the text where it is requested; (2) shape parameters are cleared at the start of a paragraph, so how are they to be reset if a new paragraph, possibly preceded by vertical space, is begun partway down the figure; (3) *T_EX* assumes a math display will last for 3 lines ([103]†), but if a math display occurring to the side of the figure takes more than 3 lines (perhaps the figure is intended to illuminate the long calculation), how are the shape parameters to be reset correctly after the display.

One approach to the first problem is, to let the figures "float" to wherever there is room for them. The `\topinsert` and `\midinsert` commands of plain *T_EX* can be used to place figures in this way, although of course those commands do not try to surround the figure with automatically shaped paragraphs. One says a figure inserted that way floats in the main vertical list. Thomas Reid wrote macros (*TUGboat* 8, no. 3, p. 315) which, instead of using insertions, place figures at the right while letting them float in the list of paragraphs. His macros were extended and adapted for *L_AT_EX* by Thomas Kneser (*TUGboat* 12, no. 1, p. 28). They do shape the text surrounding the figure, and they provide a solution of problem (2), at least for paragraphs which are separated by normal `\parskip` glue. But when the typesetting is finished, if a figure which is supposed to clarify or is repeatedly referred to in a portion of text has floated off some distance away from it, the reader may find such separation inconvenient.

At any rate, in the scheme given here, the figure always goes exactly where it was placed in relation to the text. For example, the figure anchored to the left (and it is at the left) was set there by the command `\leftopic` (defined below), given at the start of this paragraph. The solution to the first problem is simply to break the page just above the figure, if there really is no room on that page for the figure. The white space this creates can be erased, when revisions are done, simply by moving the figure command a bit.

The *T_EXbook* offers solutions to the second problem in Exercises 14.23 and 14.24. The first idea given there is to combine paragraphs by the side of the figure into a single one, which is shaped to fit the figure and then reformed into the original arrangement of paragraphs. The other idea is to use `\prevgraf` to count the lines that remain to be indented at the start of each new paragraph. However, these approaches are not really suited for use by macros which try to place figures automatically. For one thing, they would require that some unusual instructions be given to start each paragraph, or vertical skip and paragraph, by the side of a figure—you couldn't just leave a blank line or say `\medskip`, etc. Indeed, the first approach would require the whole document

initially to be a single paragraph, if there are figures all through it.

But the main difficulty is that the number of lines a figure takes up—that is, the vertical extent of the figure divided by `\baselineskip` (in compatible units)—will not be the number of lines to indent if there are vertical skips between some of the lines, while the solutions to the exercises only take number of lines into account. If for instance, by way of experiment, you say '`\hangindent=1in \hangafter=-3`' at

† Following David Salomon, we use '[xyz]' to refer to page xyz of *The T_EXbook*.

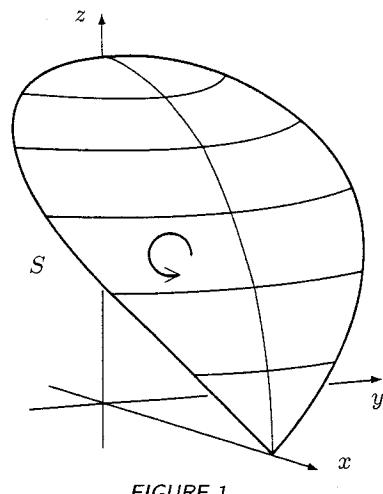


FIGURE 1

the start of a paragraph and put `\vadjust{\vskip5in}` in the second line, the typeset copy will have the first 3 lines indented an inch, but there will be a large gap between the second and third lines. Also, for this reason they don't help to solve the third problem, since as mentioned TeX doesn't directly keep track of the number of lines actually used by a math display. However, rather than counting lines, the macros given here employ `\everypar` to issue instructions at the start of each paragraph, which will freshly shape the paragraph according to where its head happens to lie in relation to the figure (Reid's method does this, too); and math displays are measured correctly by employing `\everydisplay` to arrange matters so that the remainder of the paragraph following the display is treated as a new paragraph and is shaped properly.

2 Summary of commands

Of the several macros described in the next section only four are actually used as commands when one sets figures. They are,

```
\leftpic, \rightpic, \morelines, \pixitem... \done.
```

The first two of these take as argument the name of a file that contains instructions to draw a figure, and are used to set that figure respectively to the left or right of the surrounding text. `\morelines` is used to change the number of indented lines of text surrounding a figure (this is seldom necessary), while `\pixitem` replaces the `\item` macro of plain TeX in such text.

Definitions and usage of these commands appear below.

3 The macros

To get them going we must allocate registers and set some parameters (see figure 3 below; the "gap" values are of course subject to change as desired):

```
\newcount \spts   \spts=\baselineskip          % convert <dimen> to <number>
\newcount \lines  \newcount \moreline \newcount \pp \newcount \qq
\newdimen \breadth \newdimen \length \newdimen \cobreath
\newdimen \overgap \overgap=2\baselineskip      % space above the figure
\newdimen \sidegap \sidegap=35pt                 % space to the side
\newdimen \undergap \undergap=3\baselineskip     % space below, if there's room
\newdimen \drop   \drop=-1pt
\newbox\picbox  \newbox\bottom \newskip\prskp \prskp=\parskip
\newif\ifright \newif\ifroom \newif\ifshort \newif\ifflag \newif\ifmore
\newif\ifeqo \newif\ifleqo
```

Here `\drop` is a quantity which, once the setting of a figure has begun, measures the distance from the top of the page to the bottom of the figure, including space allowance at the bottom; but it needs to be less than zero at the start of a page, as we'll soon see. So it is initialized to `-1pt`, and is reset to that value page by page, as part of the output routine.

```
\everypar={\checkpic} \def\emptypar{\everypar={}} \everydisplay={\adjustdisplaylines}
\output={\fancyoutput}
\def\fancyoutput{ % When short is true,
  \ifshort\unvbox255\setbox\bottom=\lastbox % look at the bottom line:
  \ifdim\wd\bottom>\cobreath
    \penalty-201\box\bottom % if it's too long, break before it.
  \else\ifvoid\bottom\else\restoreleftindent
    \ifdim\wd\bottom<\cobreath\parskip=0pt\noindent\vadjust{\penalty-1}%
    \ifleqo\global\leqofalse % Is the bottom line a displayed formula?
    \else\advance\cobreath-\wd\bottom % if there's no \leqno, restore its indent;
    \ifeqo\global\eqofalse\else\divide\cobreath2\fi\hskip\cobreath\fi\fi
    \box\bottom\restorepenalty % finally put the line back.
  \fi\fi
  \global\holdinginserts=0\global\shortfalse
  \else\plainoutput\global\drop=-1pt\global\onpenalty\fi} % when short is false
\def\offpenalty{\widowpenalty=1\clubpenalty=2\brokenpenalty=1}
\def\onpenalty{\widowpenalty=150\clubpenalty=150\brokenpenalty=100}
\def\restorepenalty{\ifnum\outputpenalty=10000\else\penalty\outputpenalty\fi}
\def\restoreleftindent{\ifright\else\parskip=0pt\noindent
  \vadjust{\penalty-1}\hskip\breadth\fi}
\let\eqo=\eqno \let\leqo=\leqno \def\eqno{\ifshort\global\eqottrue\fi\eqo}
```

```
\def\leqno{\ifshort\global\leqottrue\fi\leqo} \def\lzip{\phantom7}
```

`\checkpic` (in `\everypar`) will essentially compare `\drop` and `\pagetotal` at the start of each paragraph, in order to see whether paragraph shaping is needed at that point. If `\pagetotal` $\geq \drop$, it must be that the head of the paragraph lies below the figure, so shaping is not needed; otherwise, the paragraph shape parameters are set to fit the remainder of the figure. `\pagetotal` is zero when a page begins ([114]), hence if `\drop < 0` then, a new paragraph at page top will not be shaped to fit some figure which was placed on the previous page. This accounts for some of the last part of `\fancyoutput`; we'll get back to the rest later, but usually it all reduces to `\plainoutput`, unless the figure is being set close to the bottom of the page.

If you are putting other tokens into `\everypar`, then your list should include `\checkpic` when these macros are being used. By the same token you need to keep `\adjustdisplaylines` on your `\everydisplay` list.

3.1 Reading in figures. I use `PICTEX` to draw my figures: it happens to be just right for my "chalkboard" style diagrams, such as figure 2 at the right, and has the obvious advantages that go with its being part of `TeX`, while its disadvantages of time and memory requirements are tending to evaporate as equipment improves and titanic `TeX`s come along; and I put the `PICture` instructions in a separate file which is then input to the main `TeX` file. Accordingly, these macros expect to see the name of a file which, when read in by means of `\setbox\picbox=\hbox{\input{file name}}`, will cause the figure to be drawn on the page when one says '`\box\picbox`'; or '`\unhbox\picbox`', as given below in `\setpic`, since `PICture` instructions already define an `hbox`. (The space after the file name keeps the '`}`' out of the name. `\input` itself does not use curly braces.) Naturally, a different way of producing figures may call for modifications, but the macros as they are will set a figure as long as it is found in an `hbox` in `\box\picbox`. The perceived size of the figure will be the size of the `hbox`. One handy feature of `PICTEX` is that '`\hbox{\input{file name}}`' produces a box in which the `PICture` just fits; if the figure is created differently you may need to specify the box dimensions. (By the way, if you are using `PICTEX`—these macros intentionally ignore the location of the "reference point" assigned by the `\setcoordinatesystem` command of `PICTEX`, or at least the location of the point in relation to the `PICture` as `TeX` sees it.)

To set the figure described by a file named `picfoo.tex` at the left margin, to the left of surrounding text, you say '`\leftpic{picfoo}`'; on the other hand, '`\rightpic{picfoo}`' will set the figure to the right. For instance, figure 1 above was drawn by the file `pictug2.tex` and figure 2 was drawn by `pictug3.tex`, and the figures were inserted by the commands '`\leftpic{pictug2}`' and '`\rightpic{pictug3}`', respectively. The text following the figure command will then be placed around the figure, meaning that a paragraph will be started and shaped to fit the box containing the figure; the command ends the paragraph of text preceding it. A blank line following the figure command makes no difference.

```
\def\leftpic#1{\rightfalse\putpic{#1}} % file named #1 draws the figure
\def\rightpic#1{\righttrue\putpic{#1}}
\def\putpic#1{\par\previous\stall
  \setbox\picbox=\hbox{\input#1}} % now \box\picbox holds the figure
  \measure
  \ifroom\setpic
  \else\toss\measure\stall\setpic\fi}
\def\previous{\progress\ifnum\dimen7>0 % take care about the previous figure: if
  \ifshort\shortfalse\toss % it's 'short,' go now to the next page;
  \else\kern\dimen7\fi\fi} % otherwise leave enough space.
\def\progress{\dimen7=\drop\ifflag\advance\dimen7-\dimen3\flagfalse
  \else\advance\dimen7-\pagetotal\fi} % set \dimen7 by progress alongside figure
\def\toss{\vfill\eject}
\def\stall{\pp=\pageshrink\divide\pp\spts\advance\pp1 \qq=\pp \parskip=0pt
  \loop\ifnum\pp>0 \line{}\advance\pp-1 \repeat\kern-\qq\baselineskip\penalty0}
```

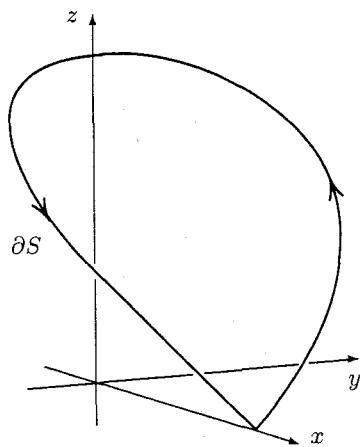


FIGURE 2

% in case the figure is just past page bottom, or at page top

The `\par` which starts `\putpic` is supposed to terminate horizontal mode, thus you should not use these figure commands inside an `hbox` ([286]). After the figure is packed into `\box\picbox`, the figure box is measured, by `\measure`, and is set on the page by `\setpic`, if there is room for it. But if there happens to be no room on the current page, in the sense of problem (1), the page is ended just above the figure and the figure is set at the top of the next page.

The `\previous` macro has the job of keeping things orderly when there are several figures marching in close succession; `\progress` essentially sets `\dimen7` equal to `\drop` minus `\pagetotal`; `\stalling` is necessary in case a figure command is encountered just past what will be the bottom of a page. We'll get back to those macros eventually.

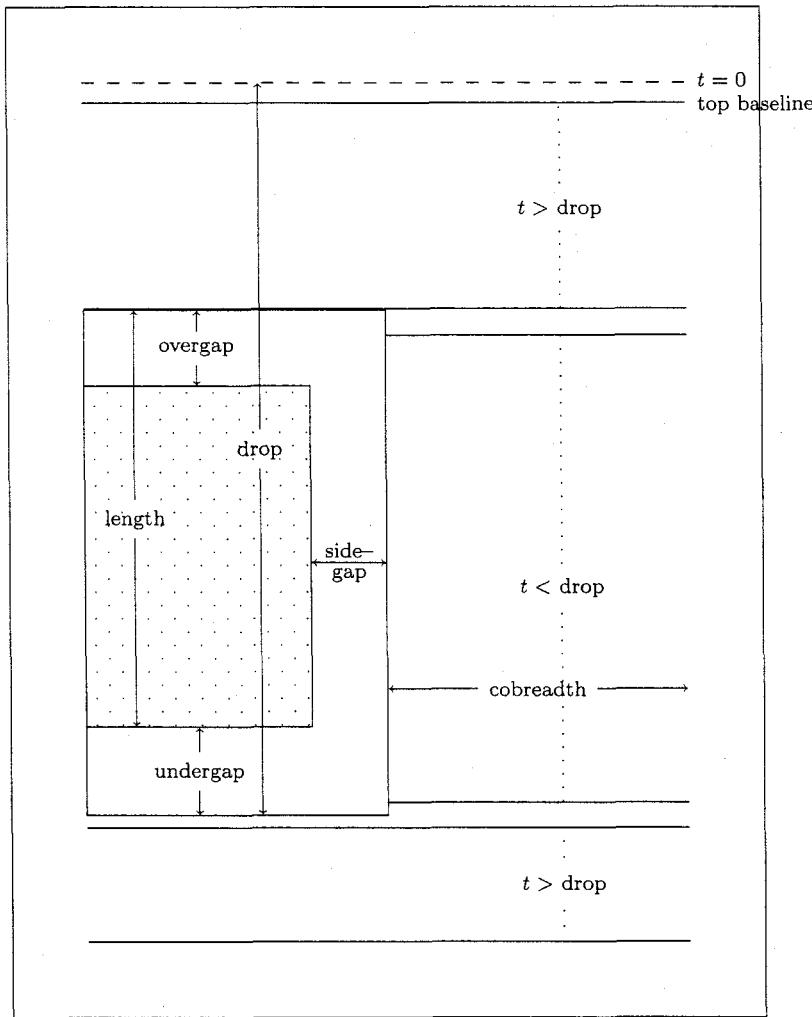


FIGURE 3

3.2 Measure the figure box. `\measure` is mode-independent; it just measures things, sets `\drop`, etc., and makes some decisions.

```
\def\measure{\flagtrue\dimen3=\pagetotal
\breadth=\wd\picbox\advance\breadth\sidegap
\cobreadth=\hsize\advance\cobreadth-\breadth      % correct width of indented text
\length=\ht\picbox\advance\length\dp\picbox\advance\length\overgap
```

```
\drop=\length\advance\drop\dimen3
\ifdim\drop>.99\pagegoal\roomfalse
\else\dimen0=\pagegoal\advance\dimen0-\drop
\ifdim\dimen0<\undergap\advance\drop\dimen0
\shorttrue\offpenalty\holdinginserts1
\else\advance\drop\undergap\shortfalse\fi
\roomtrue\fi}

% no room on this page
% there is room
% short space at the bottom
% [400]
% plenty of space at the bottom
```

The ‘flag’ being true will just mean, in the `\progress` macro, that paragraph shaping is going to occur right at the start of a figure, rather than partway down; the current value of `\pagetotal` is saved for later reference. The box containing the figure is measured, and the measurements are then increased, on top by `\overgap` and on the side by `\sidegap`.

If the bottom of that so far augmented box would extend past the nominal page bottom, as determined by the size of `\pagegoal`, we set `\room` false, which will tell `\putpic` to toss the present page and hustle to the next one. Otherwise we allow some space beneath the box: if there is plenty of room we allow `\undergap`, but if doing that would exceed the `\pagegoal` we leave just enough space to reach the nominal page bottom; in this last case we say that the figure, or the gap beneath it, is “short.”

In the illustration at the right, the space left beneath the figure box is very narrow since the figure comes nearly to the bottom of the page.

3.3 Place the figure.

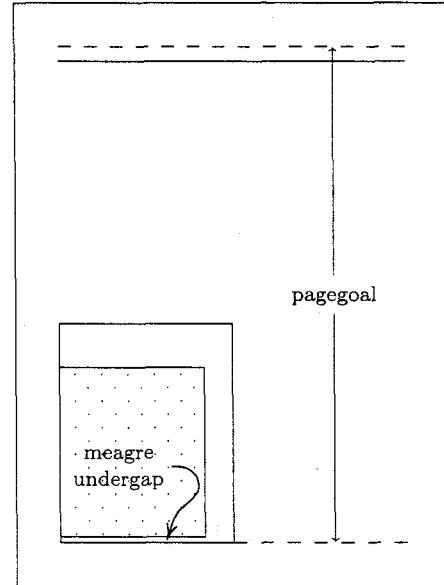
Sooner or later the figure gets set on the page.

```
\def\setpic{\kern\overgap
% \stall preserves the \kern at page top
\dimen5=\prevdepth\offinterlineskip
% save the depth of the previous line
{\emptypar\parskip=0pt\ifright\hfill\unhbox\picbox % empty \everypar inside the group
\else\noindent\unhbox\picbox\hfill\fi}%
\par\kern-\length\penalty10001 % go back up; don't break now!
\normalbaselines\prevdepth=\dimen5\ignorespaces}
% the next paragraph invokes \checkpic
```

The basic plan here is to place the figure at one margin or the other, then hop back up† and continue with the text following the command. But there is some joinery required. If the figure is at page top, as in figure 5, the first line of indented text should be at the normal position for the first line on a page. Otherwise, the first

indented line should be the correct distance below the line of text just preceding the figure. And, in both cases the top of the figure should be in the same location relative to the first line of indented text.

Well—actually, `\setpic` does the first two things, and ordinarily it does the third, but a first indented line containing a tall character will be lower than ordinary in relation to the figure. In fact, the top of the figure box is placed the distance `\overgap` below the baseline of the box of text just above the figure; if the figure is at page top, an empty `\line{}`, one `\baselineskip` higher than normal, is substituted for the box



A "Short" Figure

FIGURE 4

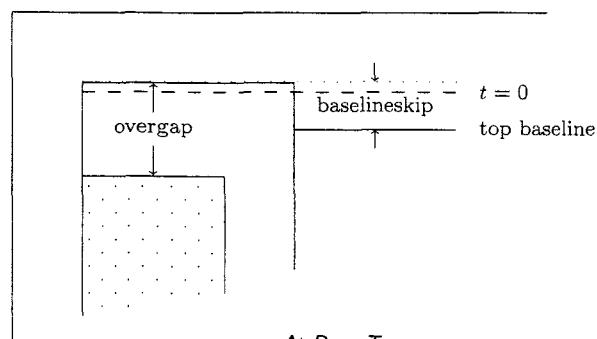


FIGURE 5

† I am obliged to Karl Berry for suggesting this idea.

above. That line, which keeps TeX from discarding the `\overgap` kern at the top of the page, will be set there by `\stall`, which we'll get to later.

TeX will want to start a new paragraph when it has to set the figure box, so the effects, vertical and otherwise of its doing so are temporarily annulled (which means the reference point of a Picture gets ignored, as mentioned above). When the figure box is in place and we are back up again, the value of `\prevdepth` is restored so that the first indented line will be set properly.

3.4 Shaping. But how do the lines get indented. We exit `\setpic`, hence the figure command, in vertical mode. Recall that `\everpar` contains `\checkpic`:

```
\def\checkpic{\progress\ifnum\dimen7>0 \shapepar{fi}\parskip=\prskip}
\def\shapepar{\lines=\dimen7 % convert <dimen> to <number>
\divide\lines\spts
\ifnum\lines>0
\ifright\hangindent=-\breadth\else\hangindent=\breadth\fi
\hangafter=-\lines
\ifmore\advance\hangafter-\moreline\morefalse\fi\fi}
```

and `\checkpic` will if necessary summon `\shapepar` to shape the lines in the paragraph following the figure command, that is unless `\pagetotal` \geq `\drop` as we said earlier. When the figure is being set on the page, the value of `\pagetotal` is changed a little by `\setpic`, and the amount of change depends on whether the figure is at page top or not; but this change is ignored by `\progress`, hence by `\checkpic` and `\shapepar`, instead they all use the original value of `\pagetotal` saved as `\dimen3`. Once the figure is in place, `\flag` is turned off and they just use the current `\pagetotal` value. Then, the amount of indentation and the number of lines to be indented are reckoned, according to current conditions. (We have to exclude the case that `\lines=0`, since TeX will indent the entire paragraph if `\hangafter=0`.)

It may sometimes happen that a paragraph by the side of a figure begins with text which is inside a group, for instance so as to be slanted when typeset. `\checkpic` will be inserted at the beginning of the paragraph, but then TeX will forget that the paragraph was shaped to fit the figure once the group ends, so in that case you need to say `\checkpic` following the group. For instance, the fourth manuscript paragraph of this article begins with the TUGboat macro `\TB`, which sets 'The TeXbook' in slanted type, and continues with '`\checkpic` offers ...'

3.5 Ending math displays; getting more lines. The last line of `\shapepar` comes into use if you want to change the number of indented lines next to the figure, perhaps to accommodate a math display that ends or begins near the bottom of the figure, or to compensate for vertical glue directly following a figure command (where `\checkpic` will not see it).

```
\def\morelines{\global\moretrue\global\moreline} % put _before_ a closing $$
\def\adjustdisplaylines#1$${\#1$\$ \par\vskip-\parskip\noindent\ignorespaces}
% the next paragraph invokes \checkpic
```

To get one more indented line in the last of the indented paragraphs next to a figure, you say '`\morelines1`', or '`\morelines=1`', just before the start of that paragraph, that is before the `\everypar` tokens for the paragraph have been inserted; '`\morelines-2`' will give you two fewer indented lines, etc.

`\everydisplay` contains `\adjustdisplaylines`; this macro takes the displayed math material as argument for its delimited parameter, the delimiter being the `$$` which closes the math display, and the macro repeats the displayed material and the closing `$$`. Then the text immediately following the display (if there is any) is put into a new paragraph, unparindented and unparskipped but shaped properly by `\checkpic` and `\shapepar`. Thus `\morelines`, if it is to be used to alter the number of lines indented in that paragraph, needs to go in *before* the closing `$$` (hence the '`\globals`' in its definition), since the paragraph shape parameters will already have been set when material after the closing `$$` is being read.

This use of `\everydisplay` follows a suggestion of Stephan von Bechtolsheim. It is convenient in that displays are enclosed by pairs of `$$`s in customary fashion, but because the displayed material is treated as a macro argument, occasional problems can arise. For instance, if you use a construction such as `$$\vbox{\settabs{+}{sample line}\cr...}$$`, TeX will complain because `{+` is `\outer`; replacing `{+` with `\tabalign` doesn't fix things because the `\settabs` will then expect to see a number of columns ([355]). One way out is locally to redefine `{+` as `\tabalign`, thus erasing its `\outerness`. Another approach is to define a replacement for the closing `$$` of a display, say by `\def\enddisplay{$$\par\vskip-\parskip\noindent\ignorespaces}`; then the displayed material will not be a macro argument. However, in that case you

have to say `\enddisplay` to close a display (instead of `$$`) if you want the text immediately following it to be shaped to fit some recent figure.

If the text to be indented by the side of the figure happens to contain a `\vbox` or a `\vtop` in which the `\hsize` has been lessened, you should probably put `\emptytypar` (defined as `\everypar={}`) at the start of the material in the box; otherwise TeX will try to leave room for the figure inside the box. The effect of `\emptytypar` is confined to the group in the box. For instance, in the following sample, `$$\eqalign{S_j &= \vtop{\emptytypar\hsize=130pt \{n-1\}-box...}\cr ...}$$` was used next to the figure.

... Let $S = \partial K$. S consists of n pairs of facing sides. Let S_j, S'_j denote the j^{th} pair, where

S_j = the $n-1$ -box which is spanned
at P by all the v 's except v_j ;

S'_j = the parallel translate of S_j to
the head Q_j of v_j .

S_j and S'_j are $n-1$ -boxes, and are oriented as part of the boundary of the n -box K . S_j is spanned at P by v_i 's as we indicated, but their increasing order may not be the positive order on S_j ; likewise S'_j is spanned by the parallel translates of the v_i 's to Q_j , though not necessarily in that order ...

3.6 The output routine. About that `\output` routine. It went like so:

```
\def\fancyoutput{ % When short is true,
  \ifshort\unvbox255\setbox\bottom=\lastbox % look at the bottom line:
    \ifdim\wd\bottom>\cobreath % if it's too long, break before it.
    \penalty-201\box\bottom
    \else\ifvoid\bottom\else\restoreleftindent
    \ifdim\wd\bottom<\cobreath\parskip=0pt\noindent\vadjust{\penalty-1}%
      \ifleqo\global\leqofalse % Is the bottom line a displayed formula?
      \else\advance\cobreath-\wd\bottom % if there's no \leqno, restore its indent;
      \ifeqo\global\eqofalse\else\divide\cobreath2\fi\hskip\cobreath\fi\fi
      \box\bottom\restorepenalty % finally put the line back.
    \fi\fi
    \global\holdinginserts=0\global\shortfalse
  \else\plainoutput\global\drop=-1pt\global\onpenalty\fi % when short is false
  \def\offpenalty{\widowpenalty=1\clubpenalty=2\brokenpenalty=1}
  \def\onpenalty{\widowpenalty=150\clubpenalty=150\brokenpenalty=100}
  \def\restorepenalty{\ifnum\outputpenalty=10000\else\penalty\outputpenalty\fi}
  \def\restoreleftindent{\ifright\else\parskip=0pt\noindent
    \vadjust{\penalty-1}\hskip\breadth\fi}
  \let\eqo=\eqno \let\leqo=\leqno \def\eqnof{\ifshort\global\eqottrue\fi\eqo}
  \def\leqnof{\ifshort\global\eqottrue\fi\leqo} \def\lzip{\phantom7}
```

If `\short` is false then this is just `\plainoutput`, along with the resetting of `\drop`. Otherwise—Recall that `\short` is set true by `\measure` when the distance from the bottom of the figure box to the nominal pagebottom, as given by the value of `\pagegoal`, is positive but less than `\undergap`. In that case the number of indented lines is based on the location of the nominal pagebottom, and the page break is expected to occur at or just before that point. But the page builder might decide to break the page a line later than expected, with the result that the last line on the page will be full width, or at least wider than `\cobreath` if it is a wide displayed formula; it will not be indented and will probably run over part of the figure, or its caption. This will not do, so we have to persuade the page builder to make the right page break. As `\short` is true, the `\output` routine will on its first pass look at what TeX thinks the last line should be. If in fact that line is wider than `\cobreath`, a negative penalty is inserted before it, to encourage the desired breaking at the penalty item, that is just before the line; otherwise the line is put back where it was, re-indented as necessary (see below); then the page is reprocessed.

More exactly, in the absence of penalties, the cost c of a page break is (according to the page-breaking algorithm, [111]) just the badness b when $b < 10000$, that is when the `\pagetotal` t is near the `\pagegoal` g .

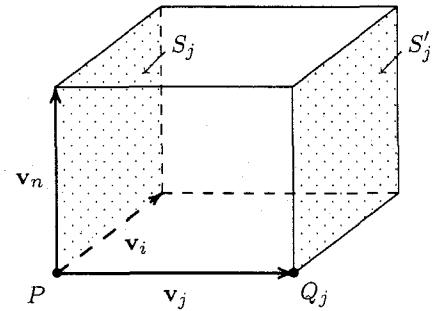


FIGURE 6

Hence c is a decreasing function of t when t is close to but less than g , and it is an increasing function when t is close to but greater than g . (Roughly, $c \approx |t - g|^3$ times a factor which depends inversely on the amount by which the page can stretch (if $t < g$) or shrink (if $t > g$), [77, 97, 111].) Of the two least values of c for t on either side of g , sometimes one will be less, sometimes the other; but each of them will be less than all the other costs on its side, so one of those two corresponding lines will be selected as the place to break. Only the higher line on the page, that is the line with $t < g$, will be indented (by `\shapepar`), the next line will be full length. When the cost of breaking after the lower line would in fact be less, a penalty is inserted by the `\output` routine to discourage that break, as we said. (The `\penalty-201` used here is sufficient for pages of ordinary length, but a more negative penalty would be necessary for pages of small `\vsize`, since when b changes on such pages it tends to do so very suddenly.)

If, however, there were club, widow or broken penalties attached to some of the lines around the breakpoint, all this reasoning would be susceptible of failure, so it is necessary to suppress those penalties for the time being; this is done by placing `\offpenalty` in `\measure`, in the part where `\short` is set true. (The penalties are here set to 1 or 2 instead of 0 just so that they will continue to show up in diagnostic output from `\tracingthings`. You may prefer that they = 0.)

Of course, penalties which are suppressed will be unable to give the protection they were designed for, and you may get club lines, etc., in consequence. Furthermore, other penalties which affect page breaking remain unsuppressed here. For instance, a math display near the bottom of a figure near the bottom of a page may drag the short line preceding it, along with it to the top of the next page (and perhaps cause an underfull page), because of the `\predisplaypenalty`. Or the text may end with a `\bye` just past a page that contains a figure at the bottom and some shrinkable glue, such as appears when there are displayed equations ([189], [348]); then `TEX` will get to read the `\bye`, and the `\supereject` penalty issued by the `\bye` may force the last lines of text to be part of that page, so that they run over the lower portion of the figure.

But in such cases it is probably better to fix by rewriting.

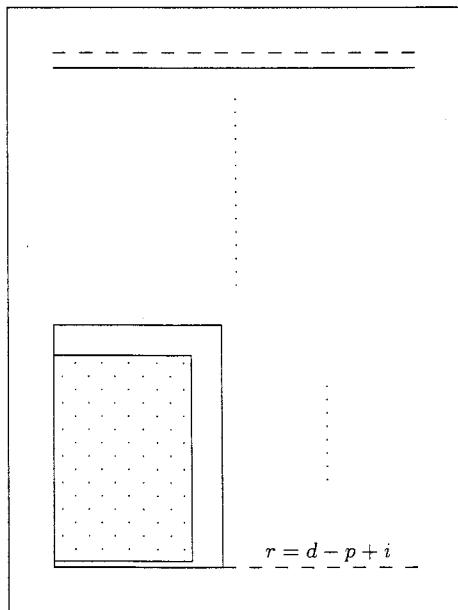


FIGURE 7

When the higher line is going to be chosen for the last line on the page, we want to put the contents of `\box255` back onto the main vertical list so as to achieve a “seamless join between the completed page and the subsequent material” ([254]), thus if the breakpoint is a penalty item we should reinsert it; and in any case we shall have to hold insertions in place during the first pass through `\output` (see [125] and [254]). In `TEX 3`, `\outputpenalty=10000` unless the breakpoint is a penalty item ([125]), so we only restore it in the latter case. Insertions are held in place by setting `\holdinginserts=1` in `\measure`, in the part where `\short` is set true, then they are released in time for the second pass. The point of holding them is to be sure that `TEX` leaves enough room during the second pass, otherwise `\pagegoal` will be reset to full size and the insertions will be crammed onto an already full page, with an overfull page as the distressing result.

Sometimes the last line on a page has to be re-indented. Assume that `\short` is true and the length of the last line is less than `\cbobreadth`. The case that `\lastbox` be void is excluded,[†] so on its first pass the `\output` routine will set `\box\bottom` to an hbox containing the line, and will then put that hbox back onto the main vertical list, where the last line was. But any instruction there may originally have been to shift the line will be lost in this process; it will not be saved in `\box\bottom`. On its second pass, the output

[†] `\lastbox` will be void for instance on the final page of text, when the last item on the vertical list is the `\supereject` penalty issued by a `\bye`.

routine is `\plainoutout` and, if we do nothing, it will place that line unshifted, flush left; so we may be obliged to re-indent the line by ourselves. In fact, this will be necessary if `\leftpic` is being used, or if the last line is a displayed formula, as in figure 7. In the first case, we insert, with `\vadjust`, a small penalty item following the shifted line, so as to permit the page break after the line. In the second case the correct indentation will depend on whether there are equation numbers or not, according to the rules for positioning equation numbers ([188–189]). Also, in that case there is a complication: a displayed formula with a restored indent is followed on the vertical list by an infinite penalty, which prevents the desired page break. To remedy this we again insert an ameliorating penalty item following the line, so as to permit the break after all.

It could happen that `\short` is true and there are several displayed formulas next to the figure, some of which have equation numbers put there by `\eqno` or by `\leqno` (we assume that the numbers are either all at the right or all at the left), and some with no numbers. In that case, for the sake of correct indentation, it is necessary to supply bogus equation numbers to formulas without real ones. You can do this by typing '`$$<formula>\eqno$$'` or '`$$<formula>\leqno\lzip$$'` for those formulas, depending on the circumstances.

There is one possibility which is overlooked by our re-indenting scheme, namely that in which the last line is a displayed formula with an equation number, and its actual width is less than `\cbobreadth`, but the line is considered to be below the figure, not an indented line. Then it belongs on the next page but it will not be put there by `\fancyoutput`, and the spacing for the equation number will be too wide. However, an application of `\morelines` is an effective, if not an automatic, fix for this.

3.7 The previous figure. When there are more figures than text, the natural separation by prose may not keep one figure from treading on another, so we must make arrangements to avoid overlaps. The `\previous` macro does the job:

```
\def\previous{\progress\ifnum\dimen7>0      % take care about the previous figure: if
  \ifshort\shortfalse\toss                  % it's 'short,' go now to the next page;
  \else\kern\dimen7\fi\fi}                  % otherwise leave enough space.
```

Recall that `\dimen7` is set by `\progress`, essentially to `\drop - \pagetotal`. If `\dimen7 > 0` when a figure command is read, text is still being set by the side of the previous figure. If the previous figure on this page was short, we certainly don't want to try to fit the current figure on the same page, so it is tossed to the next page. As the last line will then be empty there is no need to take a close look at it, thus `\short` is set false. But ordinarily a figure is not short, and when it is not, at least the `\undergap` space is kept between its bottom and the top of the `\overgap` space above the next figure (as illustrated just below).

If you want to bring successive figures closer together than the current `\undergap + \overgap`, the way to do it is to decrease the size of `\undergap`, before the start of the upper figure, or of `\overgap`, before the lower figure. A negative `\vskip` just before the second figure command will be ignored, since `\progress` will take the skip into account when it computes `\dimen7`, and leave that much more vertical space.

3.8 Stalling. Why is it we have to `\stall`, at the beginning of `\putpic`?

```
\def\stall{\pp=\pageshrink\divide\pp\spts
  \advance\pp1 \qq=\pp \parskip=0pt
  \loop\ifnum\pp>0 \line{}\advance\pp-1
  \repeat\kern-\qq\baselineskip\penalty0}
```

This maneuver becomes necessary when a figure command turns up just past what will be the eventual breakpoint on a page that contains some shrinkable glue. In that case, the page builder has *TEX* continue to read lines until it finds one of infinite cost, but if there is shrinkable glue that could well be several lines past the best breakpoint; and by then *TEX* may have encountered and begun reading the figure command. Big mess! With `\pagetotal > \pagegoal` by that point, `\measure` will find no room at all on the current page for the figure, so `\putpic` will issue

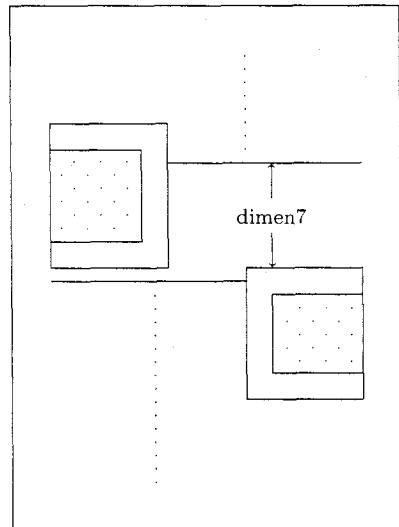


FIGURE 8

`\vfill\eject`, meaning to toss the figure to the next page; but by the time everything is sorted out there may be a very short page where the figure was supposed to go, with the figure itself on the page after that. Under such circumstances we must stall, and issue a series of junk `\line{}`s to prevent TeX's continuing to read the command, until the page builder has caught up in its exercise and decided on the breakpoint for the page, and has initiated the `\output` routine. The number of junk lines issued depends on the amount of shrinkable glue on the page. `\stall` ends with `\penalty0` so that the effect of the negative `\kern` which cancels the junk lines may be recorded in `\pagetotal`.

The number of junk lines issued is always at least one. This conveniently serves a different purpose than that just mentioned, namely to set a figure correctly when it has to go at page top: the junk line preceding the figure saves the `\overgap` kern at the start of `\setpic` from being discarded in that case.

Of course, `\stalling` is superfluous for most figures; but in a document of any length, and subject to revision, you never know when you'll need it.

3.9 One more item. Plain TeX provides the macro `\item` ([102], [355]), which allows one to typeset numbered, indented items. This macro can be useful in connection with figures, for instance if the figure illustrates several notions which you want to itemize by its side; but `\item` achieves hanging indentation by setting `\hangindent=\parindent`, so we cannot use it in combination with our figure commands since they need `\hangindent` themselves. However, we can get the same result using `\leftskip`.

```
\def\xitem#1 {\par\removelastskip\smallskip\noindent\llap{\#1\enspace}%
  \ignorespaces\leftskip=0.75\parindent\begingroup\parindent=0.5\parindent}
\def\done{\endgroup\par\smallskip\leftskip=0pt}
% use like \item; end with \done
```

The parameter has the same usage as `\item`'s parameter, except it is delimited by a space. You can have more than one paragraph in each item, but to finish each item you need to say '`\done`'. The indentation is diminished since there will be less room than normal next to the figure. Items are set off by `\smallskip`s, fore and aft.

The pair of `\pars` surrounding the `\endgroup` in the definition of `\done` is necessary because of the grouping ([100]). The first `\par` allows the final paragraph inside the group to be shaped properly, the second does the same for the paragraph following the group. Without the first `\par`, the group would end before its final inside paragraph, and that paragraph would be shaped according to the values that `\hangindent` and `\hangafter` had at the start of the group, but with `\leftskip=0pt`. With the first `\par` but without the second, the final inside paragraph would have `\leftskip=0.75\parindent`, but now the text following the group would be shaped according to the values that `\hangindent` and `\hangafter` had in the final inside paragraph.

◇ Daniel Comenetz
46 Burnham Street
Belmont, MA 02178

The \CASE and \FIND macros

Jonathan Fine

Abstract

This article is a continuation of the author's *Some Basic Control Macros for T_EX* in *TUGboat* 13, no. 1. It introduces macros \CASE and \FIND which are useful for selecting an action to be performed on the basis of the value of a parameter. These macros cannot be used in the mouth of T_EX. Also, some changes to the *Basic Control Macros* are reported.

Introduction

First an example is given of the use of the \CASE macro, and then the macro itself is given. The next section does the same, for the \FIND macro. On both occasions, step-by-step examples of the functioning of these macros are given. A discussion of pitfalls in the use of the macros follows, and some other items, and finally a report on the *Basic Control Macros* is given. To the best of my knowledge, there are no jokes in this article.

Much of the inspiration for \CASE and \FIND came from studying Mittelbach and Schöpf's article *A new font selection scheme for T_EX macro packages — the basic macros* in *TUGboat* 10, no. 2, while the rest came from the author's own needs.

Independently, Kees van der Laan has developed a macro \loc which has something in common with \FIND. It can be found on page 229 of his article *FIFO and LIFO incognito*, which appears in the *EuroT_EX 92* proceedings, published by the Czechoslovak TUG.

In about 1,000 lines of documented code the author had occasion to use \continue (and the ':' variant) 17 times, \return 5 times, and \break but once. The macro \CASE was used 5 times, and \FIND twice. By comparison, the 17 primitive \if... commands of T_EX were used 35 times altogether.

Acknowledgements The author thanks the referee for many helpful comments, and particularly for requesting a more accessible exposition.

1 The \CASE macro

The \CASE macro is similar to the \SWITCH macro defined in *Basic Control*. However, it requires assignment and so cannot be used in the mouth of T_EX.

By way of an example, suppose that one wishes to code a macro \fruit such that

```
\fruit\a produces apple
\fruit\b produces banana
\fruit\x produces \error\x
```

where \error is to handle unknown arguments to the \fruit command.

To code the macro \fruit, the association of the *(key)*s \a, \b, etc., with the *(action)*s apple, banana, \error\x must be stored in some form or another. Using the semicolon ';' as a delimiter, the code fragment

```
\a apple ;
\b banana ;
\x \error\x ;
```

will store the data for the \fruit macro. Each of the lines above we will call an *(alternative)*.

The \fruit macro can be coded as

```
\def\fruit #1
{
  \CASE #1
    \a apple ;
    \b banana ;
    % default action
    % omit at your own peril
    #1 \error #1 ;
  \END
}
```

where the macro \CASE is to search for the token #1 amongst the *(key)*s and then extract and execute the associated *(action)*. (This and all other code in this article is assumed to be read in an environment where white space characters are ignored. I will explain later how to set this up.)

Here is the code for the \CASE macro which supports this style of programming.

```
\long\def\CASE #1
{
  \long
  \def\next ##1           % discard
    ;#1                 % find <key>
    ##2;                % <action>
    ##3 \END            % discard
    { ##2 }              % copy <action>
  \next ; % do \next - note the ';'
```

Perhaps the easiest way of understanding the \CASE macro is to follow its functioning step by step. We shall do two examples, \fruit\b and \fruit\x. The example of \fruit\aa—which is left to the reader—shows why the ';' is required after \next in the definition of \CASE.

The expansion of \fruit \b is

\CASE \b \a apple;\b banana;\b \error \b ;\END
and now \CASE expands to produce

```
\long \def \next #1;\b #2;\#3\END {#2}
\next ;\a apple;\b banana;\b \error \b ;\END
```

and so \next will be defined as a \long macro with ;\b and ; and \END as delimiters. After \next has been defined the tokens

\next ;\a apple;\b banana;\b \error \b ;\END
remain. Now for the crucial step. By virtue of the definition of \next, all tokens up to the \END will be absorbed while forming the parameter text for \next. The tokens ;\a apple form #1. The vital parameter #2 is banana. Finally, #3 get \b \error \b;. Thus, the result of the expansion of \next will be

banana

just as desired. (The T_EXbook discusses macros with delimited parameters on pages 203-4.)

Now for \fruit \x. The first level expansion will be

\CASE \x \a apple;\b banana;\x \error \x ;\END
where the #1 in the default option has been replaced by \x. As before, \CASE \x will define \next to be delimited by ;\x and ; and \END. This time, because \x is not an explicit key within \fruit, the default *(action)*

\error \x

will be the result of the expansion of \next. As mentioned earlier, \error is to handle unknown arguments to \fruit.

This last example shows the importance of coding a default option within a \CASE. This option should be placed last amongst the *(alternative)s*. If omitted an unknown key will cause the scratch macro \next to not properly find its delimiters. Usually, this will result in a T_EX error.

2 The \FIND macro

There are situations—for example the problem of printing vowels in boldface—where several of the values of the parameter will give rise to what is basically the same action. The \FIND macros is better than \CASE in such situations.

Suppose that the desired syntax is that

\markvowels Audacious \end

is intended to produce

Audacious

where \end is used as delimiter.

The macro \markvowels can be coded as a loop, reading tokens one at a time. It is to be concluded when \end is read. Should the token read by \markvowels be a vowel, then this letter should be printed in boldface, otherwise the token should be printed in the default font. Vowel or

not, \markvowels is a loop and so after processing a non-\end token \markvowels should be called again.

Thus, there are three sorts of actions

- print token in \bf and continue
- print token in default font and continue
- end the loop—i.e. do nothing

and as any of the ten letters aeiouAEIOU give rise to the first type of action, it is better to use \FIND, which is similar to \CASE except that a single alternative can have several keys.

The syntax for \CASE is

```
\CASE <search token>
  % one or more times
  <key> <option> ;
  % don't forget the default
\END
```

while for \FIND the syntax is

```
\FIND <search token>
  % one or more times
  % one or more <key>s
  <key> ... <key> * <option> ;
  % don't forget the default
\END
```

where \FIND will look for the *<search token>* (amongst the *<key>*s, we hope) and having found it will save the *<option>* (between * and ;) as it gobbles to the \END, and then execute the *<option>*.

So much for the theory. We shall now code the macros \markvowels and \FIND, and then run through some examples step by step. Here is the emboldening macro coded.

```
\def\markvowels #1
{
  \FIND #1
    % the <action> for \end is empty
    \end * ;

    % vowels
    aeiou AEIOU
    * {\bf #1} \markvowels ;

    % other tokens
    #1 * #1 \markvowels ;

  \END
}
```

where \FIND should search for the *<key>* and then the next * tag. What follows up to the next ; is the selected *<action>*, which is to be reproduced. The remaining tokens up to \END are discarded.

```
\long\def\FIND #1
{
  \long
  \def\next ##1          % discard
    #1      % find <key>
    ##2 *   % discard up to
              % next tag
    ##3 ;   % <action>
    ##4 \END % discard
    { ##3 }   % copy <action>
  \next           % do \next
}
```

Now for the examples. We shall follow the expansion of `\markvowels AZ\end`, step by step.

First, `\markvowels A` will expand to yield

```
\FIND A\end *;aeiouAEIOU*\{bf A}\markvowels ;
A*A\markvowels ;\END Z\end
```

(please note the `Z\end` awaiting processing after the `\END`) and now `\FIND` expands

```
\def \next #1#2##3:#4\END {#3}\next
  \end *;aeiouAEIOU*\{bf A}\markvowels ;
  A*A\markvowels ;\END Z\end
```

to define `\next` delimited by `A * ; \END`. The expansion

```
\next
  \end *;aeiouAEIOU*\{bf A}\markvowels ;
  A*A\markvowels ;\END Z\end
```

of `\next` will result in

```
{bf A}\markvowels Z\end
```

and so the letter `A` will be set in `\bf`. The tokens `Z\end` have been carried along, from the beginning of this example. Next, `\markvowels` is expanded

```
\FIND Z\end *;aeiouAEIOU*\{bf Z}\markvowels ;
Z*Z\markvowels ;\END \end
```

and as before `\FIND` results in

```
\long \def \next #1#2##3:#4\END {#3}\next
  \end *;aeiouAEIOU*\{bf Z}\markvowels ;
  Z*Z\markvowels ;\END \end
```

the definition of `\next` (delimiters `Z * ; \END`), whose expansion

```
\next
  \end *;aeiouAEIOU*\{bf Z}\markvowels ;
  Z*Z\markvowels ;\END \end
```

produces

```
Z\markvowels \end
```

and so `Z` is set in the default font. Now for `\markvowels \end`, which expands to

```
\FIND \end \end *
  aeiouAEIOU*\{bf \end }\markvowels ;
  \end *\end \markvowels ;\END
```

and again `\FIND` defines `\next`

```
\def \next #1\end #2##3:#4\END {#3}\next
  \end *;aeiouAEIOU*\{bf \end }\markvowels ;
  \end *\end \markvowels ;\END
```

(with delimiters `\end * ; \END`) and the expansion

```
\next
  \end *;aeiouAEIOU*\{bf \end }\markvowels ;
  \end *\end \markvowels ;\END
```

of `\next` is empty. (Why is this? The macro `\next` will first search for `\end`. The tokens before this `\end` form `#1`. They happen to be empty, but in any case they are discarded. Similarly, `#2` is empty, and is discarded. However, `#3` is the `<action>`, and in this case it is empty. The remaining tokens, between `\end * ;` and `\END`, form `#4`, and are discarded.)

(In terms of `\FIND`, the `\loc` macro of van der Laan can be written as

```
\def\loc #1#2
{
  \FIND #1
  #2 * \let\iffound\iffalse ;
  #1 * \let\iffound\iftrue ;
  \END
}
```

but there is no easy expression for `\FIND` in terms of `\loc`.)

3 Warnings

There are several ways in which these macros can trip up the unwary.

No default A default action must be supplied, and it should be the last option, unless you are certain that it will never be required. The code fragment

```
\lowercase{ \CASE #1 }
  h \help ;
  #1 ;
\END
```

lacks a default, for when `#1` is `A` the fragment

```
\CASE a
  h \help ;
  A ;
\END
```

remains once `\lowercase` has executed. To avoid this, either apply `\lowercase` to the *whole* `\CASE` statement, or write

```
\lowercase{ \amacro #1 }
```

where `\amacro` contains the `\CASE` statement.

Meaning ignored The `\CASE` and `\FIND` macros depend on the token passed as parameter, but *not* on its `\meaning`. This token can be a control sequence or a character token. Thus, the operation

of `\markvowels` is independent of the meaning of `\end`. This is often what is wanted, but is different from usual `\ifx` comparison.

Braces stripped Selecting an option such as

```
\group * { \bf stuff } ;
```

within `\FIND` will result in

```
\bf stuff
```

being processed without the enclosing braces—an error which nearly occurs in `\markvowels`. This is a consequence of TeX's rules for reading parameters. The same failure can happen with the `\CASE` macro.

Braces not supplied Consider the macro

```
\def\puzzle #1
{
  \FIND #1
    abc * [#1] ;
    def * (#1) ;
    #1 * '#1' ;
  \END
}
```

applied to `x`. The result of `\puzzle x` will not be the default '`x`'. It will be '(`x`)'!

The invocation of `\FIND x` will produce

```
\long\def\next #1 x #2 * #3 ; # 4 \END
{#3}
```

and as `x` will replace `#1` in `\puzzle`, the parameters to `\next` will be (delimiters italicized)

```
#1 <- abc * [ x
#2 <- ] ; def *
#3 <- (x) ;
#4 <- x * 'x' \END
```

and so in this situation the `(action)` for `def` will have been selected.

The problem is that `#1` is prematurely visible. The solution is to hide it. This is done by writing

```
abc * { [#1] } ;
def * { (#1) } ;
```

which has enclosed the troublesome `(action)s` in braces. As mentioned earlier, these braces will be stripped before the action is executed.

Surplus semicolons Code such as

```
\FIND #1
  0123456789
  * \action\one ;
  \action\two ;
  #1 * \default #1 ;
\END
```

is deceptive. When the parameter is 1 only `\action\one` will be performed. (There is an

erroneous semicolon that the eye easily misses.) In this context the layout

```
\FIND #1
  0123456789
  *
  \action\one
  \action\two
;
#1 * \default #1 ;
\END
```

reads better.

4 Setting up the catcodes

The macros `\CASE` and `\FIND` will have confusing results if the characters ; or * are passed as parameters. This may happen if the document author writes `\fruit;` or `\markvowels Abc; def \end.` To prevent this confusion while preserving the syntax we shall alter some catcodes. We shall also ignore white space. By setting

<code>\catcode'\"=4</code>	<code>\catcode'\"*=4</code>
<code>\catcode'\' =9</code>	<code>\catcode'\"^I=9</code>
<code>\catcode'\"^M=9</code>	<code>\catcode'\"^=10</code>

at the beginning of the file containing `\CASE` and `\FIND`, and macros calling `\CASE` and `\FIND`, and placing

<code>\catcode'\";=12</code>	<code>\catcode'\"*=12</code>
<code>\catcode'\' =10</code>	<code>\catcode'\"^I=5</code>
<code>\catcode'\"^M=10</code>	<code>\catcode'\"^=13</code>

to restore values at the end of the file, we can be sure that any ; or * characters generated by a document author will not match the private delimiting tokens ; and * used within `\CASE`, `\FIND`, and their calling macros.

The character ~ has been given a `\catcode` of 10 which is `(space)`. According to *The TeXbook*, p47, when a character with `\catcode` `(space)` is read from a file, it is "converted to a token of category 10 whose character code is 32" and so ~ can be used to place an *ordinary* space token into a macro. Incidentally, it is a consequence of this rule, and the rules for `\uppercase`, `\lowercase`, and `\string` (see pages 40–41) that it is impossible to place a character token with category 10 and character code zero into the stomach of TeX.

(The characters ; and * have been given `\catcode` 4, which is `(alignment tab)`, to help detect errors. If the TeX error message

! Misplaced alignment tab character ;.

or similar with * occurs, then there is an error in the coding or execution of a \CASE or \FIND macro.)

5 Variable delimiter macros

The macros \CASE and \FIND are particular examples of what I call *variable delimiter macros*. They are useful for control and selection. Their essence is to define and execute a scratch macro—\next—which has as delimiter a token that was originally passed as a parameter.

Even though TEX is fixed and unchanging, change can be discussed. Currently a macro parameter character # cannot serve as a delimiter for a macro. The code

```
\def\#{}##{}
```

will produce the TEX error

```
! Parameters must be numbered
      consecutively.
```

and this provides a place for an extension to be built.

Suppose that ## were allowed in the parameter text of a macro, to stand for a variable delimiter. Then \FIND could be coded as

```
\long\def\FIND #1 ## #2 * #3 ; #4 \END
{ #3 }
```

where the expansion of \FIND consists of first replacing ## by the next token in the input stream (assumed not { or } or #) and then expanding the resulting macro.

The code in this style

```
\long\def\CASE #1 ; ## #2 ; #3 \END
{ #2 }
```

for \CASE is not quite right, for it misses the vital semicolon after \next in the original definition.

6 Benefits of the \noname package

The catcode changes listed above—or rather the effect of these changes—is obtained automatically should the macro file be processed by the author's \noname package, which is described in *TUGboat* 13, no. 4. Should the macro writer wish to place an *ordinary* ; or * within a \CASE or \FIND macro, this can easily be done using \noname. (Without \noname this will require explicit and unpleasant dirty tricks.)

The \noname package will also translate the label ':' used by the *Basic Control* macros into an otherwise inaccessible control sequence, as it \loads a macro source file.

The step-by-step expansion of examples of the use of the \CASE and \FIND macros was generated

by the single-step debugger \ssd which is also part of the \noname package. (The output has been lightly edited to improve the appearance, and particularly to get decent line breaks.)

7 Basic Control — a report

Experience has brought the following changes to the basic control macros.

In the original article, both ':' and \END were used as delimiting labels. It turns out to be more convenient to have but one label. Thus one has

```
\long\def\break #1 : #2 {}
\long\def\continue #1 : {}
\long\def\chain #1 #2 : #3 { #1 }
\long\def\return #1 #2 : { #1 }
```

and the \fi'ed variants, but \exit has gone and \return gobbles to ':' rather than to \END. Another change—the macros are now \long.

Finally, the *soft double-fi*

```
\def\:::fi { \fi \fi }
```

is introduced for the situations where one would like to have \:::continue, etc., available. (Just write \:::\continuer instead.)

The macro \switch has so far turned out to be not so useful. Much of its functionality has been subsumed by \CASE and \FIND.

◊ Jonathan Fine
203 Coldhams Lane
Cambridge
CB1 3HY
England
J.Fine@pmms.cam.ac.uk

Using TeX to make Agendas and Calendars with Astronomical Events

Jordi Saludes

At the Cork TeX meeting participants were presented with an agenda obviously typeset with TeX. Calendars and agendas show a lot of computed information, like the number of day in the month, the name of day in the week. TeX is well-suited to deal with these computations. Agendas also give relevant events for the user: 'Fixed' events occurring every year at fixed dates but also 'movable' events which depend on astronomical phenomena related to the sun and moon, such as religious feast days or moon phases.

It would be nice to have a package to typeset agendas including fixed events, even better, to have TeX compute the date of movable events and include them in the agenda.* We describe here a macro package to make calendars and agendas for a given year, supporting

- Easy inclusion of custom events;
- Automatic inclusion of computed events.

This package is divided into several files devoted to computation of different type of events. File `calend0.tex` is the basic macro file; it provides tools to convert between and compute date formats. File `calend1.tex` gives macros to support computation of astronomical events which are calculated in `sun.tex` and `moon.tex` and file `feasts.tex` allows computation of movable feasts based on the date of Easter Sunday. We provide several example style files containing formatting instructions to typeset agendas and calendars.

Acknowledgments. The author is grateful to B. Beeton and the reviewer for suggesting several style formats.

Basic Mechanism

Events. To obtain an agenda for year *(my year)* you have to create a source file including custom events

`\event{day}/{month}[\text]`

in order the agenda will show *(text)* in the cell corresponding to *(month)*, *(day)*.

The prototypical source file looks like this:

```
\input calend0
\setyear{my year}
\settimezone{my timezone}
<my definitions>
\begin{events}
\input moon
\input sun
\input feasts
\event 6/6[Mom's anniversary]
\event 5/1[TUG Meeting]
<more events>
:
\includemoons
\includeseasons
\includefeasts
\end{events}
```

And the procedure is as follows:

1. Run TeX for the first time. This process reads the events and puts them into an auxiliary event file *(jobname).eve*. Each `\event` generates an entry in *(jobname).eve* converting the calendar date of event to long date

`\evententry{long date}{text}`.

For `\setyear{1993}` and `\settimezone{0}` the event file looks like

```
\evententry{34125}{Mom's anniversary}
\evententry{33973}{TUG Meeting}
```

:

```
\evententry{33969}{First quarter}
\evententry{33976}{Full Moon}
```

:

```
\evententry{34047}{Spring}
```

:

```
\evententry{34069}{Easter Sunday}
```

:

2. Sort the lines of the auxiliary file using some sorting utility. Now events are in chronological order.

3. Add a `\makeagenda{style}` command just before `\begin{events}` and run TeX for the second time. This process reads in the auxiliary file and generates an agenda or calendar including the events in suitable days (see appendix A).

Converting dates.

- The *long date* is the Julian date referred to 1900 Jan 0, i.e. the number of days passed

* Like the `\primes` example of [K1], p. 218, TeX not only writes, but provides the information too.

from 1899 Dec 31 up to current day. This package uses the long date as the base date to `\evententry` commands and to compute the day of the week.

- Precise astronomical computations require not only the date but also the fraction of day passed. We define *Julian date and time* (JDT) as the number of days passed from 1900 Jan 0.5 (i.e. 1899 Dec 31 at noon), including fractions of day.

Command `\dayno` performs the conversion between ordinary date contained in counters `\Day` and `\Month` and year in `\setyear` to long date. The value is returned in `\date`. This is used, for instance, when processing `\event` which is expanded to `\dayno` plus `\evday` which generates the actual entry in the event file. `\evday{\text{}}` is like `\event` but reads the long date of the event from counter `\date`. Note that `\text{}` is not expanded in `\event`, while it is in `\evday`; this is because `\evday` is intended for automatic generation of events whose description is, in general, the product of a macro expansion.

To convert between long dates and usual calendar dates, use `\caldate`. This will transform the value of `\date` to `\Day` and `\Month`; `\themonth` gives the name of month number `\Month`. The week day is given by the remainder of the integer division of long date by 7: `\weekday` does it for counter `\date`, returning the value to `\weekd` in such a way that Sunday is 0 and Saturday is 6. `\theweekday` gives the name of the day. `\theshortweekday` gives an abbreviation of the week day. Both are language-dependent. In the files of this package you can search for the string ‘local’ to find commands which depend on language or other local assumptions.

Command `\nextday{number}` advances counter `\date` to the next day with `\weekd={number}`; `\prevday` works like `\nextday` but going backwards. There are strict versions of these commands: `\snextday` and `\sprevday`. The difference between the strict and normal ones is the same as between ‘<’ and ‘ \leq ’; `\snextday` (`\sprevday`) always changes `\date` by at least 1 day, while `\nextday` (`\prevday`) does not. For instance, assume `\date` is the long date of a Monday then `\snextday1` advances `\date` by 7, but `\nextday1` leaves `\date` unchanged. These macros are useful when computing movable feasts which have to fall in a given week day.

JDT is based on ephemeris time which is no more than 2 minutes from UT (Universal time). We

can then assume that $UT \approx JT + 0.5$, where JT is the fractional part of JDT. Command `\JDTtoL` converts JDT to local long date by adding the value of your timezone and discarding fractions of day. You can state your timezone using `\settimezone{\text{}}`, where `\text{}` is the difference in hours between your time and Greenwich time (West negative, East positive).

When a computed event requiring JDT falls near local midnight it is probable that, due to approximation errors, truncation would place the event in the adjacent day.

Style files

The `\makeagenda{\text{}}` command in the second TeX run reads the style file `\text{.sty}` and accomodates the events following the formatting instructions contained in this file. We provide example files for two agenda styles and two calendars.

All these examples support marking of holidays by including `\holy` in the description part of an event. The style file can then use `\ifholy` to test this condition when making the cell day. In the example files, the expresion `\ifholy\bf\fi` is used to typeset in bold the day number of a holyday.

A style file has to define the fonts being used, set the dimension parameters, and state the period of time covered by the agenda by setting counter `\currentdate` to the initial long date and `\lastdate` to the long date of the last day of the wanted period. This is how `\makeagenda` generates the cells:

- S1. The cell for the first day (actual value of `\currentdate`) is opened: the calendar date of `\currentdate` is placed into `\Day` and `\Month` and `\beginday` is executed. This control sequence is defined in the style file as instructions for making the day cell up to the moment when the events for this day have to appear.
- S2. Now an `\evententry` is read. If the date of the event d is not $\currentdate \leq d \leq \lastdate$, the entry is ignored.
- S3. If $d = \currentdate$ and the cell day for `\currentdate` is opened the event text is added as a paragraph (If the cell for `\currentdate` was not opened yet, it is done now). Back to S2.
- S4. If $d > \currentdate$ the open cell is closed by doing `\endday`. This control (symetric to `\beginday`) defines how to finish a cell day properly.
- S5. The value of `\currentdate` is incremented and a cell without events is generated. This is

repeated until $d = \text{\currentdate}$. Then jump to S3.

When all the events have been read, cells without events are created if necessary to reach \lastdate . Now \TeX expands \epilog which contains instructions to be executed after the last day cell was typeset. \epilog must always be defined though it can have an empty expansion.

The control sequence \makeempty generates day cells for dates between \currentdate and \lastdate but without including any event.

A5 style. File **a5.sty** contains A5 two-page style that gives space (cells) for each day separated by rules. Each cell displays the \Day for the month number and the day of the week. Events (if any) are listed right down the rule. The cell is of fixed height, so you can run over into the following cell if you have too many events in a day. This style gives one page per week and each page begins with a monday. The footline displays the name of the current month. Note that the agenda begins at the first Monday of September and finishes the Sunday after the end of May, to give a sort of academic-year agenda.

For styles like this one, not in a single natural year, you have to state events twice, since \includemoons , \includeseasons and \includefeasts generate events for one year. For example, the following will give moons for years 1992 and 1993

```
\setyear{1992}
\begin{events}
<92-events>
\includemoons
\nextyear
<93-events>
\includemoons
\endevents
```

A6 style. This style gives a double-page per week agenda for a natural year. Saturday and Sunday get half size cells. The headline of the left page contains the number of the week in the year while the other displays the current month. Sundays and holidays show a bold day numer. The style file is **a6.sty**.

Wall style. Put $\text{\Month}=\langle my\ month\ number \rangle$ before $\text{\makeagenda\{wall\}}$ to obtain an A4-landscape wall calendar for this month. Days are distributed in a grid, each week (Sunday to Saturday) in a row.

If you forget to include the \Month assignment, the style typesets the current month if today belongs to the first half of the month, otherwise it gives the next month.

Block style. Desktop block calendar (1 page per day). Each page displays the month name, a big day number and the week day. Small calendar displays of the current and next months appear on the left and right side of the big number. Note the use of \makeempty in the style file **block.sty**.

Astronomical computations

Complex numerical routines are done without allocation of new counters; \count0 to \count8 are used instead. To prevent interference with the standard use of these registers in the output routine, calculations remain local by grouping them into braces. The returned value is extracted using a global assignment.

Arithmetic. There are two kinds of computations needed

- Integer arithmetic computations;
- Computations involving trigonometric functions.

Approximation errors and overflow limit the validity of computations to years between 1900 and 2100.

Integer arithmetic is the natural one in \TeX . Calculations can be done using the primitive commands \advance , \multiply and \divide . Remember that (as stated in [K1], p. 118) the last operation means integer division (discarding the remainder) for positive values; for negative numbers this operation is equivalent to division of the absolute value and changing of sign. Conversions between dates are done in integer arithmetic.

The second class of computations is handled by a fixed-point approach and approximate calculation. A fractional value a is stored in a counter as \tilde{a} such that $a = \tilde{a} \cdot 10^{-n}$ and \tilde{a} is an integer with $|\tilde{a}| < 2^{31}$. We say that a is scaled 10^{-n} ; for example, JDT is usually stored in counter \date as $\text{\JDT} = \text{\date} \cdot 10^{-3}$. Thus, we can express JDT with an accuracy of $10^{-3} \approx 1.44$ minutes.

The astronomical formulae we use require products of high precision numbers (more than 6 digits) but we can't multiply two such numbers using single \TeX counters. In this case we store a in blocks a_0, \dots, a_k of m -digit integers such that $a = (a_0 + a_1 10^m + a_2 10^{2m} + \dots + a_k 10^{km}) \cdot 10^{-n}$. For instance $\text{\lin}(a_2) \cdot (a_1) \cdot (a_0) + (b)$ computes $\theta = aT + b$, where $T = \text{\count0} \cdot 10^{-6}$ is Julian centuries and a is a 9-digit precision factor divided in 3 blocks (a_2 , a_1 and a_0) to avoid overflow during multiplication; b and θ are scaled 10^{-3} .

Accuracy conditions on the formulae imply that trigonometric functions have to be computed

with a maximum error of 0.01. To evaluate $\sin \theta$ the argument is reduced modulo 2π and the value is computed by linear interpolation on a table with step 0.1 in the interval $[0, \pi]$ (following [H], this is sufficient to fit in the above error). For $\cos \theta$ we use the equivalent form $\sin(\pi/2 - \theta)$.

The moon. File `moon.tex` allows generation of events for full, new moon and quarters in the given year. Formulae [M], pp. 159–161, imply computation of a sum of trigonometric functions evaluated with arguments which are linear transformations of T (the Julian century of the moment). We have adapted these formulae by discarding high order terms keeping an accuracy of 45 minutes.

Counter `\moonno` stands for the number of quarters of moon cycle passed from the beginning of 1900. `\moonno` modulo 4 gives the type of quarter as follows

Phase	<code>\moonno</code> (4)
New moon	0
First quarter	1
Full moon	2
Last quarter	3

The command `\includemoons` generates events for all quarters in the given year. First, the approximate value of `\moonno` is computed for the first phase in the year, then `\moondate` gives the exact long date and generates an `\evday` entry. This is performed repeatedly until the end of year is reached. Text for `\evday` comes from `\phase` (language dependent).

The sun. The beginning of seasons is determined by the dates of March and September equinoxes and June and December solstices. We use formulae [M], p. 90 (neglecting quadratic terms), that give values in JDT with error generally less than $1/2$ hour. These events are generated by `\includeseasons` in file `sun.tex`.

... and all together. Finally the file `feasts.tex` allows the inclusion of movable feasts based on the date of Easter Sunday, which is intended to be the first Sunday after the first full moon of spring. However the ecclesiastical computation does not always coincide with that definition since it assumes that the moon moves uniformly, so we take the Clavius-Lilius algorithm (XVI Century) used by most Western churches (see [K2]), instead of deriving the date from astronomical data.

File `feasts.tex` gives examples using `\easter` and `\snextday` in the computation of some movable religious feasts as in [E]. For instance, note that (in some countries) Epiphany is defined as the first

Sunday after New Year's day. Use `\includefeasts` to put feast dates into the events file.

References

- [K1] D. E. Knuth, *The TeXbook, Computers & Typesetting* vol. A. Addison-Wesley (1984).
- [K2] D. E. Knuth, *The Art of Computer Programming: Fundamental Algorithms*. Addison-Wesley (1978).
- [M] J. Meeus, *Astronomical Formulae for calculators*. Willmann-Bell, Richmond (1985).
- [H] P. Henrici, *Elements of Numerical Analysis*. John Wiley & Sons, New York (1984).
- [E] Éphémérides 1984. Annuaire du Bureau des Longitudes, Gauthier-Villars, Paris (1983).

◊ Jordi Saludes
ETSEIT, Universitat
Politècnica de Catalunya
Colom, 11
08222 Terrassa, Spain
`saludes@ma2.upc.es`

Appendix A. Example pages

Examples are given of the following styles:

- Block style
- A5 style
- Wall style
- A6 style

Except for the wall style, a faint rule has been drawn around the example pages to indicate their borders.

A.1 Block style

JANUARY

A.2 A5 style

	Monday 6
	Tuesday 7
	Wednesday 8
Last quarter	Thursday 9
	Friday 10
	Saturday 11
	Sunday 12

September

A.3 Wall style

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
27	28	29	30	31	1 First quarter	2
3 Epiphany	4	5 TUG Meeting	6	7	8 Full Moon	9
10	11	12	13	14	15 Last quarter	16
17	18	19	20	21	22 New moon	23
24	25	26	27	28	29	30 First
31	1	2	3	4	5	6 Full M

A.4 A6 style

Week #2

January

4 Mon			Thu 7
5 Tue	TUG Meeting Full Moon		Fri 8
6 Wed		Sat 9	Sun 10

Appendix B. The macros

B.1 calend0.tex

```
%%
%% FILE calend0.tex
%% Modificat 9/12/92
\catcode`\@=11
\newif\ifleapyear
\def\loadadvanced{%
  \input calend1.tex\relax}
\newcount\date
\newcount\weekd
\newcount\Year
\newcount\yearbase
\newcount\Month
\newcount\Day
\newcount\@catch
\newcount\timezone\timezone=0
\def\setyear#1{\Year=#1
  \advance\Year by-1900\calculateyearbase}
\def\nextyear{\advance\Year by1
  \calculateyearbase}
\def\settimezone#1{\timezone=#1
  \multiply\timezone by 1000
  \divide\timezone by24}
% \ifleapyear is set;
% \yearbase is the number of
% days passed from 1900, Jan 0
% to New year's date;
\def\calculateyearbase{%
  \yearbase=-1
  {\count0=\Year\divide\count0 by4
  \multiply\count0 by4
  \ifnum\Year=\count0
  \global\@catch=0\else\global\@catch=1\fi}%
  \ifcase\@catch \leapyeartrue\or
  \leapyearfalse\fi
  {\count0=\Year\multiply\count0
  by1461 \advance\count0 by3
  \divide\count0 by4
  \global\@catch=\count0}%
  \advance\yearbase by\@catch
  \ifnum\Year=0\yearbase=0\leapyearfalse\fi}
% Gives the number of days passed
% at the end of each month.
% Value returned in \Day
\def\monthdays{\global
  \@catch=\ifcase\Month 0\or31\or
  59\or90\or120\or151\or181\or212\or
  243\or273\or304\or334\or365\fi
  {\ifleapyear\ifnum\Month>1
  \global\advance\@catch by1\fi\fi}%
  \Day=\@catch}
% Long date of \Month, \Day
% in the year \Year.
% Value returned in \date
\def\dayno{\date=\Day\%}

\advance\Month by-1\monthdays
\advance\date by\Day
\global\advance\date by\yearbase}
% Long date MOD 7 gives the week day.
% Sunday is 0 and Saturday is 6.
\def\weekday{{\count0=\date\relax
  \count1=\count0\divide\count0 by 7
  \multiply\count0 by 7
  \advance\count1 by -\count0
  \global\weekd=\count1}}
% Name of week day (Local).
\def\theweekday{\weekday
\ifcase\weekd
Sunday\or Monday\or Tuesday\or
Wednesday\or Thursday\or Friday\or
Saturday\fi}
% (Local)
\def\theshortweekday{\weekday
\ifcase\weekd Sun\or Mon\or Tue\or
Wed\or Thu\or Fri\or Sat\fi}
% Gives the usual calendar date for
% a long date in counter \date.
% Returned in \Day, \Month and \Year
\def\caldate{\Year=\date
\multiply\Year by4\divide\Year by1461
\calculateyearbase
{\advance\date by-\yearbase\Month=0
\loop\monthdays\ifnum\Day<\date
\global\advance\Month by1\repeat
{\advance\Month by-1\monthdays
\advance\date by-\Day
\global\Day=\date}}}
% Name of month \Month. (Local)
\def\themonth{\ifcase\Month
\or January\or February\or March\or
April\or May\or June\or July\or
August\or September\or October\or
November\or December\fi}
% Find the next (previous) day after
% (before) \date with \weekday=#1.
\def\nextday#1{\{\count0=#1\weekday
\advance\count0 by-\weekd
\ifnum\count0<0\advance\count0 by7\fi
\global\advance\date by\count0\}}
\def\prevday#1{\snextday#1
\advance\date by-7}
\def\snextday#1{\advance\date by1
\nextday#1}
\def\sprevday#1{\advance\date by-1
\prevday#1}
% \begin{events}... \end{events}
% contains control sequences like
% \event or \evday or sequences
% generating these commands.
\newwrite\evefile
\def\begin{events}{%
\immediate\openout
\evefile=\jobname.eve\relax}
\def\end{events}{\immediate\closeout}
```

```

\evefile\end}
\newtoks\evetext
\def\event#1/#2[#3]{\evetext={#3}%
\Day=#1\Month=#2\dayno
\evday[\the\evetext]}
\def\evday[#1]{\immediate\write\evefile{%
\string\evententry{\the\date}{#1}}}
\def\thecaldate{\the\Day/\the\Month}
\def\mute{\def\thecaldate{}}
%
% 2nd run commands
%
\newcount\currentdate
\newcount\lastdate
\newcount\nextdate
\newif\ifdoing@day
\newif\ifholy
\def\holy{\global\holytrue}
\def\upto#1{\nextdate=#1
\loop\advance\currentdate by1\relax
\ifnum\currentdate<\nextdate
\begin@day\end@day\repeat}
\def\evententry#1#2{\unskip
\ifnum#1>\lastdate\else
\ifnum\currentdate<#1
\ifdoing@day\end@day\fi
\upto#1\fi
\ifnum\currentdate=#1
\ifdoing@day\else\begin@day\fi
#2\par\fi\fi}
\def\begin@day{\do{#1}\relax
\date=\currentdate\caldate
\message{<\thecaldate>\beginday}
\def\end@day{\endday\do{#1}\relax
\global\holyfalse\message{>}}
\def\makeagenda#1{\input #1.sty\relax
\begin@day\input\jobname.eve\relax
\evententry{\the\lastdate}{}\end@day
\epilog\end}
% Do not read events
\def\makeempty{\begin@day
\evententry{\the\lastdate}{}\end@day}
\catcode`@=12

```

B.2 a5.sty

```

%%
%% FILE a5.sty
%%
\newcount\daysinpage
\newdimen\cellheight
\vsiz=17.5cm\voffset=-0.5in
\hsize=11cm\hoffset=-0.6in
\cellheight=\vsiz
\divide\cellheight by7
\advance\cellheight by-0.4pt
\parindent=0pt
\font\rm=cmr10 at 14pt

```

```

\font\Bf=cmbx10 at 14pt
\font\rm=cmr10
\footline={\Bf\ifodd\pageno\hfill\botmark
\else\botmark\hfill\fi}
\def\newpage{\vfill\penalty-10000}
\def\beginday{\ifnum\daysinpage>0\hrule\fi
\mark{\themonth} % Outside any box
\setbox2=\vbox\bggroup
\ifodd\pageno
\rightskip=3cm plus 1fill
\else
\leftskip=3cm plus 1fill
\fi\rm}
\def\endday{\egroup
\vbox to\cellheight{\vglue2pt
\setbox0=\hbox to\hsize{%
\ifodd\pageno
\hfill\Rm{\theweekday}\ \ifholy\Bf\fi\the\Day}
\else
\Rm{\ifholy\Bf\fi\the\Day}\ \theweekday\%
\hfill\fi}
\copy0\kern-\ht0\box2\vfill}
\advance\daysinpage by1
\ifnum\daysinpage=7 \daysinpage=0
\newpage\fi}
%
% Local: from monday ...
\Day=1 \Month=9\dayno\nextday1\currentdate=\date
% Local: ... to sunday.
\nextyear\Day=31 \Month=5\dayno\nextday0
\global\lastdate=\date
\leftskip=0pt\rightskip=0pt
\daysinpage=0
\def\epilog{}


```

B.3 a6.sty

```

%%
%% FILE a6.sty
%%
\baselineskip=10pt
\newcount\daysinpage
\newdimen\cellwidth
\newdimen\cellheight
\vsiz=5in\voffset=-0.2in
\hsize=3.5in\hoffset=-0.25in
\newdimen\sphsize
\sphsize=\hsize
\divide\sphsize by2\advance\sphsize by-0.4pt
\cellheight=\vsiz
\divide\cellheight by3
\advance\cellheight by-0.4pt\parindent=0pt
\font\bf=cmbx12
\font\rm=cmr12
\font\small=cmr9
\headline={\hfill
\ifodd\pageno
\rm\xdef\ftmk{\firstmark}%

```

```

\divide\pageno by2\advance\pageno by1
Week \#\the\pageno
\else
\bf\edef\btmpk{\botmark}%
\ifx\ftmk\btmpk\else\ftmk\fi\btmpk
\fi\hfill}
\footline={}
\def\newpage{\vfill\penalty-10000}
\def\beginday{
\ifcase\daysinpage\or\hrule\or\hrule\or\or
\hrule\or\hrule\fi
\mark{\themonth} % Outside any box
\setbox1=\vbox to\cellheight\bgroup\vglue4pt
\ifnum\daysinpage>4\hsize=\sphsize\fi
\setbox3=\vbox\bgroup
\ifnum\daysinpage>2
\rightskip=1.45cm plus 1fill
\else
\leftskip=1.45cm plus 1fill\fi
\noindent\small}
\def\endday{\egroup
\setbox0=\hbox to\hsize{\rm\ifhol\bf\fi
\ifnum\daysinpage=6\bf\fi
\ifnum\daysinpage>2
\hfill{\rm\theshortweekday}
\the\Day\hskip0.5em}
\else
\f\hskip0.5em\the\Day
\rm\theshortweekday}\hfill\fi}
\copy0\kern-\ht0\vskip0pt\box3
\vfill\egroup
\advance\daysinpage by1
\ifcase\daysinpage\or\box1\or\box1\or
\box1\newpage\or\box1\or\box1\or
\setbox2=\box1\or
\hbox{\box2\vrule\box1}\newpage
\daysinpage=0\fi}
%
% Local: from monday ...
\Day=1\Month=1\dayno\prevday1\currentdate=\date
% Local: ... to sunday.
\Day=31\Month=12\dayno\nextday0\lastdate=\date
\leftskip=3pt\rightskip=3pt
\daysinpage=0
\def\epilog{}
```

B.4 wall.sty

```

%%
%% FILE wall.sty
%%
\hsize=9.5in
\vsiz=7.5in\voffset=-0.65in
\baselineskip=12pt
\newcount\daysinrow
\newdimen\cellwidth
\newdimen\cellheight
\cellheight=1.13in
```

```

\parindent=0pt
\cellwidth=\hsize
\divide\cellwidth by7
\advance\cellwidth by-0.4pt
\font\BBf=cmbx12 at 24pt
\font\Bf=cmbx10 at 20pt
\font\Rm=cmr10 at 20pt
\font\rm=cmr12 at 14pt
\font\small=cmr9
\footline={}
\def\hstrut{\hrule height0pt depth0pt
width\cellwidth}
\def\beginday{
\ifnum\daysinrow=0\setbox1=\hbox{\vrule}\fi
\setbox0=\vbox to\cellheight\bgroup
\hstrut\hsize=\cellwidth\vskip5pt
\setbox2=\vbox\bgroup\small}
\def\endday{\egroup
\hbox{\hskip0.5em\rm\ifhol\bf\fi\the\Day}
\vskip2pt\box2\vfill\egroup
\setbox1=\hbox{\unhbox1\box0\vrule}
\advance\daysinrow by1
\ifnum\daysinrow=7\box1\hrule\daysinrow=0\fi}
%
\def\advancemonth{\ifnum\Month=12
\nextyear\Month=1\else\advance\Month by1\fi}
% If no Month is given, take today's date.
\ifnum\Month=0 \Month=\month
\ifnum\day>15 \advancemonth\fi\fi
\Day=1\dayno
\topskip=0pt\hbox{}
\vfill
\centerline{\BBf\themonth}
\penalty10000
\vskip40pt minus32pt
% Local: from Sunday ...
\prevday0\currentdate=\date
\hbox{\rm\loop
\hbox to\cellwidth{\hss\theday\hss}
\unskip\ifnum\weekd<6\advance\date by1\repeat}
\penalty10000
\vskip3pt
% Local: ... to Saturday.
\Day=1 \advance\Month by1\dayno
\advance\date by-1\nextday6\lastdate=\date
\leftskip=3pt
\raggedright
\daysinrow=0
\vbox\bgroup\hrule
\def\epilog{\vfill\egroup\supereject}
```

B.5 block.sty

```
%%
%% FILE block.sty
%%
\newcount\daysinrow
\newcount\saveMonth
\font\bf=cmbx12 at 16pt
\font\Rm=cmr12 at 48pt
\font\Bf=cmbx12 at 48pt
\font\rm=cmr12
\font\small=cmr10
\footline={}
\def\card{\mute
\saveMonth=\Month
\def\beginday{
\ifnum\daysinrow=0\setbox1=\hbox{}\fi
\setbox0=\hbox to 1.3em{\hss
\ifnum\Month=\saveMonth\the\Day\fi}
\def\endday{\setbox1=\hbox{\unhbox1\box0}
\advance\daysinrow by1
\ifnum\daysinrow=7\box1\daysinrow=0\fi}
% Local: from sunday ...
\Day=1\dayno
\prevday0\currentdate=\date
\Day=1\advancemonth \dayno
% Local: ... to saturday.
\advance\date by-1 \nextday6 \lastdate=\date
\daysinrow=0
\baselineskip=9pt\small
\global\setbox3=\vbox to 0.55in{\makeempty\vss}}
%
\def\cstrut{\vrule width0pt depth0.3in}
\baselineskip=14pt
\parindent=0pt
\vsiz=5in \voffset=-0.2in
\hsiz=4in \hoffset=-0.25in
\def\advancemonth{\ifnum\Month=12
\nextyear\Month=1\else
\advance\Month by1\fi}
\def\beginday{\ifnum\Day=1
\setbox2=\box3{\advancemonth\card}\fi
\date=\currentdate\caldate
\centerline{\bf\uppercase
\expandafter{\themonth}}
\setbox0=\vbox\bgroup\noindent\rm}
\def\endday{\egroup\vskip10pt
\centerline{\cstrut\copy2\quad
\hbox to 1in{\rm\weekday}
% Local: sundays in bf.
\ifnum\weekd=0\Bf\fi\ifhol\Bf\fi
\hss\the\Day\hss\quad\copy3}
\centerline{\rm\theweekday}
\vskip20pt\box0\newpage}
\def\newpage{\vfill\penalty-10000}
%
\Day=31\Month=12\dayno\lastdate=\date
\Day=1\Month=1\dayno\currentdate=\date
\card
```

```
\leftskip=1em plus1fill\rightskip=1em plus1fill
\def\epilog{}
```

B.6 calend1.tex

```
%%
%% FILE calend1.tex
%%
\def\loadadvanced{\relax}
% Convert from Julian date and time in
% \date to long date (in local time)
\def\JDTtoL{\advance\date by500
\advance\date by\timezone
\divide\date by1000 }
%% Trigonometric functions
\def\sintable#1{\ifcase #1 0 \or100 \or199
\or296 \or389 \or479 \or565 \or644 \or717
\or783 \or841 \or891 \or932 \or964 \or985
\or997 \or1000 \or992 \or974 \or946 \or909
\or863 \or808 \or746 \or675 \or598 \or516
\or427 \or335 \or239 \or141 \or42 \or-58
\or-158\fi}
% Reduces modulo 2\pi (requires positive
% argument theta):
% theta := theta MOD 2\pi, where
% theta = count1*10^(-3)
\def\twopimod{\count2 =\count1
\divide\count2 by6284 \count3 =1853
\count4 =6283 \multiply\count3 by\count2
\multiply\count4 by\count2
\divide\count3 by10000
\advance\count3 by\count4
\advance\count1 by-\count3 }
\newif\ifsign
% v := sin(theta), where
% v = count4*10^(-3);
% theta = count1*10^(-3)
% theta is reduced MOD 2\pi to be
% 0<=theta<2\pi by \twopimod,
% then linear interpolation is performed
% using \sintable.
\def\Sin{
\ifnum\count1<0 \signtrue
\count1=-\count1 \else \signfalse\fi
\loop\ifnum\count1>6284 \twopimod\repeat
\ifnum\count1>3142
\advance\count1 by-3142
\ifsign\signfalse\else\signtrue\fi\fi
\multiply\count1 by10 \count3 =\count1
\divide\count3 by1000 \count2 =\count3
\multiply\count3 by1000
\advance\count3 by-\count1
\count5 =\sintable{\count2 }
\count4 =\count5\advance\count2 by1
\advance\count4 by -\sintable{\count2 }
\multiply\count4 by\count3
\divide\count4 by1000
\advance\count4 by\count5
```

```

\ifsign\count4 ==\count4 \fi}
% v := cos(theta), where
% v = count4*10^(-3);
% theta = count1*10^(-3).
% \Sin is evaluated on pi/2-theta.
\def\Cos{\advance\count1 by-1571
  \multiply\count1 by-1\Sin}
% Linear transformation of T giving
% theta := a*T+b, where
% T = count0*10^(-6);
% theta = count1*10^(-3);
% a = #1*10^3+#2+#3*10^(-3);
% b = #4*10^(-3)
\def\lin#1.#2.#3+#4.{\count1 =#3
  \count2 =#2 \count3 =#1
  \multiply\count1 by\count0
  \multiply\count2 by\count0
  \multiply\count3 by\count0
  \divide\count1 by1000
  \advance\count1 by\count2
  \divide\count1 by1000
  \advance\count1 by\count3
  \advance\count1 by #4}
% Accumulate value returned by a
% trigonometric function, scaled by
% factor f, into count6:
% ac := ac+f*v, where
% ac = count6*10^(-7)
% v = value of Sin or Cos =count4*10^(-3)
% f = #1*10^(-4)
\def\fac#1{\multiply\count4 by #1
  \advance\count6 by\count4 }
\def\id{\count4=\count1 } % Identity

```

B.7 moon.tex

```

%%
%% FILE moon.tex
%%
\loadadvanced
\newcount\moonno
\def\firstmoon{\moonno=\Year
  \multiply\moonno by123685
  \divide\moonno by10000
  \multiply\moonno by4 \advance\moonno by-1
  \loop\moondate\ifnum\date<\yearbase
    \advance\moonno by1 \repeat}
% Compute date for cycle quarter MOONNO
\def\moondate{\{\count0=\moonno
  \lin 202.126.369+0.\count0=\count1
  \count6=0
  \lin 0.2.319+2907.\Sin\fac{3}
  \divide\count6 by1000
  \lin 365.249.86+7593.\id\fac{1}
  \divide\count6 by10 \count7 =\count6
  \ifodd\moonno\quarters
  \else\fullornew\fi\global\date=\count7}
\JDTtoL\message{\the\date}

```

```

% Correction for full and new moon
\def\fullornew{\count6=0
\lin -393.0.0+0.
\divide\count1 by100000000
\advance\count6 by\count1
\lin 0.628.300+6269.\Sin
\multiply\count6 by\count4
\lin -7.-700.-369+ 928.\Sin\fac{-74}
\lin 0. 628. 300+ 6269.\Sin\fac{ 1734}
\lin 1. 256. 600+12539.\Sin\fac{ 21}
\lin 8. 328. 670+ 5341.\Sin\fac{-4068}
\lin 8. 538. 220+-4597.\Sin\fac{ 10}
\lin 8. 956. 970+11610.\Sin\fac{ -51}
\lin 16. 238. 589+-5526.\Sin\fac{ -4}
\lin 16. 657. 340+10682.\Sin\fac{ 161}
\lin 16. 866. 890+ 743.\Sin\fac{ 104}
\lin 17. 285. 640+16951.\Sin\fac{ 50}
\lin 17. 495. 190+ 7013.\Sin\fac{ 4}
\lin 24. 986. 10+16023.\Sin\fac{ -4}
\lin 25. 195. 560+ 6084.\Sin\fac{ -6}
\divide\count6 by10000
\advance\count7 by\count6 }
% Correction for quarters
\def\quarters{\lin -393.0.0+0.
\divide\count1 by100000000
\count6 =\count1
\lin 0.628.300+6269.
\Sin\multiply\count6 by\count4
\lin -16. -29. -40+-4413.\Sin\fac{ 40}
\lin -7.-700.-369+ 928.\Sin\fac{ -47}
\lin -7. -72. -69+ 7198.\Sin\fac{ -30}
\lin 0. 628. 300+ 6270.\Sin\fac{ 1721}
\lin 1. 256. 600+12539.\Sin\fac{ 21}
\lin 8. 328. 670+ 5341.\Sin\fac{-6280}
\lin 8. 538. 220+-4598.\Sin\fac{ 21}
\lin 8. 956. 970+11611.\Sin\fac{ -119}
\lin 16. 238. 589+-5526.\Sin\fac{ -4}
\lin 16. 657. 340+10682.\Sin\fac{ 89}
\lin 16. 866. 890+ 743.\Sin\fac{ 79}
\lin 17. 285. 640+16952.\Sin\fac{ 3}
\lin 17. 495. 190+ 7013.\Sin\fac{ 3}
\lin 24. 986. 10+16023.\Sin\fac{ -4}
\lin 25. 195. 560+ 6085.\Sin\fac{ -6}
\count8=\count6\count6=28000
\lin 628. 300. 373+ 6270.\Cos\fac{ -4}
\lin 8. 328. 670+ 5341.\Cos\fac{ 3}
\count2=\moonno
\advance\count2 by-1\divide\count2 by2
\ifodd\count2\multiply\count6 by-1 \fi
\advance\count6 by\count8
\divide\count6 by10000
\advance\count7 by\count6 }
% Events for phases.
% Uses PHASE (local).
\def\includemoons{%
\message{Including moons}
{\firstmoon
\Month=12\Day=31\dayno\advance\date by1
\count1=\date

```

```
\loop\moondate\phase\ifnum\date<\count1
  \evday[\themoon]\advance\moonno by1
\repeat}
\def\phase{{\count0=\moonno
\count1=\moonno
\divide\count0 by4 \multiply\count0 by4
\advance\count1 by-\count0 \relax
\global\edef\themoon{\ifcase\count1
  New moon\or First quarter\or
  Full Moon\or Last quarter\fi}}
```

B.8 sun.tex

```
%%
%%      FILE sun.tex
%%
\loadadvanced
\def\linY#1.#2.#3+#4.{\count0=\Year
  \multiply\count0 by1000 \count1=0
  \lin #1.#2.#3+#4.
  \global\date=\count1 }\JDTtoL
% (local)
\def\includeseasons{%
  \message{Including seasons}
  \linY 365.242.365+ 79573.\evday[Spring]
  \linY 365.241.628+172405.\evday[Summer]
  \linY 365.242.045+266014.\evday[Fall]
  \linY 365.242.756+355784.\evday[Winter]}
```

B.9 feasts.tex

```
%%
%%      FILE feasts.tex
%%
% Date of Easter for year \Year.
% Value returned in \date
\def\easter{\count0=\Year\count1=\count0
  \divide\count0 by19 \multiply\count0 by19
  \advance\count1 by-\count0
  \advance\count1 by 1 % Golden number
  \count2=\count1 \multiply\count2 by11
  \advance\count2 by18 \count0=\count2
  \divide\count0 by30 \multiply\count0 by30
  \advance\count2 by-\count0 % Epact
\ifnum\count2=25 \ifnum\count1>11
  \count2=26 \fi\fi
\ifnum\count2=24 \count2=25\fi
\advance\count2 by-44
\multiply\count2 by-1
\ifnum\count2<21 \advance\count2 by30\fi
\Month=3 \Day=0 \dayno
\global\advance\date by\count2\snextday0
% (local)
\def\includefeasts{%
  \message{Including feasts}
  \easter\evday[Easter Sunday]
  {\advance\date by49 \evday[Whitsun]}
  {\advance\date by-7 \evday[Palm Sunday]}
  {\advance\date by-46 \evday[Ash Wednesday]}
  \advance\date by-3 \evday[Carnival]}
\Month=1 \Day=1 \dayno\snextday0
\evday[Epiphany]
\event 25/12[\holy Christmas]}
```

FIFO and LIFO sing the BLUes*

Kees van der Laan

Abstract

FIFO, First-In-First-Out, and LIFO, Last-In-First-Out, are well-known techniques for handling sequences. In *TeX* macro writing they are abundant but are not easily recognized as such. *TeX* templates for FIFO and LIFO are given and their use illustrated. The relation with Knuth's *\dolist*, answer ex11.5, and *\ctest*, p.376, is given.

Keywords

FIFO, LIFO, list processing, plain *TeX*, education, macro writing.

Introduction

It started with the programming of the Tower of Hanoi in *TeX*, van der Laan (1992a). For printing each tower the general FIFO—First-In-First-Out¹—approach was considered.² In literature (and courseware) the programming of these kind of things is done differently by each author, inhibiting intelligibility. In pursuit of Wirth (1976), *TeX* templates for the FIFO (and LIFO) paradigm will hopefully improve the situation.

In this article we will see various slightly different implementations of the basic FIFO principle.

FIFO

In the sequel, I will restrict the meaning of FIFO to an input stream which is processed argument-wise. FIFO can be programmed in *TeX* as template

```
\def\fifo#1{\ifx\ofif#1\ofif\fi
  \process#1\fifo}
\def\ofif#1\fifo{\fi}
```

The *\fifo* command calls a macro *\process* that handles the individual arguments. Often you can copy *\fifo* straight out of this article, but you have to write a version of *\process* that is specific to your application.

To get the flavor.

* Earlier versions appeared in MAPS92.1 and proceedings Euro*TeX*'92. BLU is Ben Lee User of *TeXbook* fame. It makes the title sing, I hope.

¹ See Knuth (1968), section 2.2.1.

² In the Tower of Hanoi article Knuth's list datastructure was finally used—*The TeXbook*, Appendix D.2—with FIFO inherent.

Length of string. An alternative to Knuth's macro *\getlength* (*The TeXbook*, p.219) is obtained via the use of *\fifo* with

```
\newcount\length
\def\process#1{\advance\length1 }
```

Then *\fifo aap noot\ofif \number\length* yields the length 7.³

Number of asterisks. An alternative to Knuth's *\atest* (*The TeXbook*, p.375), for determining the number of asterisks, is obtained via *\fifo* with

```
\def\process#1{\if*#1\advance\acnt by1
  \fi}
```

```
\newcount\acnt
```

Then *\fifo abc*de*\ofif \number\acnt* yields the number of asterisks: 2.⁴

Vertical printing. David Salomon treats the problem of vertical printing in his courseware. Via an appropriate definition of *\process* and a suitable invocation of *\fifo* it is easily obtained.

```
\def\process#1{\hbox{#1}}
\vbox{\offinterlineskip\fifo abc\ofif}
      a
yields b.
```

Tower of Hanoi. Printing of a tower █ can be done via

```
\def\process#1{\hbox to3ex{%
  \hss\vrule width#1ex height1ex\hss}}
\vbox{\baselineskip1.1ex\fifo12\ofif}
```

Termination. For the termination of the tail recursion the same *TeX*nique as given in *The TeXbook*, p.379, in the macro *\deleterightmost*, is used. This is elaborated as *\break* in Fine (1992), in relation to termination of the loop. The idea is that when *\ofif* is encountered in the input stream, that is, when *\ifx\ofif#1...* is true, all tokens in the macro up to and including *\fifo*—the start for the next level of recursion—are gobbled by a subsequent call to *\ofif*.⁵ Because the matching *\fi* is gobbled too, this token is inserted via the replacement text of *\ofif*. This *TeX*nique is better than Kabelschacht's (1987), where the token

³ Insert *\obeyspaces* when the spaces should be counted as well.

⁴ As the reader should realize, this works correctly when there are first level asterisks *only*. For counting at all levels automatically, a more general approach is needed, see Knuth's *\ctest*, p.376.

⁵ In contrast with usual programming of recursion start with the infinite loop, and then insert the *\if... \ofif\fi*.

preceding the `\fi` is expanded after the `\fi` via the use of `\expandafter`. When this is applied the exchange occurs at each level in the recursion. It is also better than the `\let\nxt=...` TeXnicue, which is used in *The TeXbook*, for example in `\iterate`, p.219, because there are no assignments.

My first version had the two tokens after `\ifx` reversed—a cow flew by—and made me realize the non-commutativity of the *first level* arguments of TeX's conditionals. For example, `\ifx aa\empty...` differs from `\ifx\empty aa...`, and `\if\ab\aa...` from `\if\aa\ab...`, with `\def\aa{aa}`, `\def\ab{ab}`. In math, and in programming languages like Pascal, the equality relation is commutative,⁶ and no such thing as expansion comes in between. When you are not alert with respect to expansion, TeX's `\if`-s can surprise you.

The `\fifo` macro is a basic one. It allows one to proceed along a list—at least conceptually—and to apply a (user) specified process to each list element. By this approach the programming of going through a list is *separated* from the various processes to be applied to the elements.⁷ It adheres to the *separation of concerns* principle, which I consider fundamental.

The input stream is processed argument-wise, with the consequence that first level braces will be gobbled. Beware! Furthermore, no outer control sequences are allowed, nor `\par`-s. The latter can be permitted via the use of `\long\def`.

A general approach—relieved from the restrictions on the input stream: *every token* is processed until `\ofif`—is given in *The TeXbook*, answer ex11.5 (`\dolist...`) and on p.376 (`\ctest...`). After adaptation to the `\fifo` notation and to the use of macros instead of token variables, Knuth's `\dolist` comes down to

```
\def\fifo{\afterassignment\tap
 \let\nxt=
 \def\tap{\ifx\nxt\ofif\ofif\fi\process
 \nxt\fifo}
 \def\ofif#1\fifof\fi}
```

This general approach is indispensable for macro writers. My less general approach can do a lot already, for particular applications, as will be shown below. But, ... beware of its limitations.

Variations. The above `\fifo` can be seen as a template for encoding tail recursion in TeX, with arguments taken from the input stream one after another. An extension is to take two arguments from the input stream at a time, with the second argument to look ahead, via

```
\def\fifo#1#2{\process#1\ifx\ofif#2
 \ofif\fi\fifo#2}
 \def\ofif#1\ofiff\fi{}
```

Note the systematics in the use of the parameter separator in `\ofif`; here `\ofif` and in the previous macro `\fifo`, the last token of the replacement text. Although the principle of looking ahead with recursion is abundant in computer programming, a small example to illustrate its use is borrowed from Salomon: delete last character of argument. It is related to `\deleterightmost` (*The TeXbook*p.379). Effective is the following, where a second parameter for `\fifo` is introduced to look ahead, which is inserted back when starting the next recursion level

```
\def\gobblelast#1{\fifo#1\ofif}
 \def\fifo#1#2{\ifx\ofif#2\ofif\fi%
 #1\fifo#2}
 \def\ofif#1\ofif\fi{}
```

Then `\gobblelast{aap}` will yield `aa`. And what about recursion without parameters? A nice example of that is a variant implementation of Knuth's `\iterate` of the `\loop` (*The TeXbook*, p.219).

```
\def\iterate{\body\else\etareti\fi%
 \iterate}
 \def\etareti#1\iterate{\fi}
```

This `\iterate` contains only 5 tokens in contrast with Knuth's 11. The efficiency and the needed memory is determined by the number of tokens in `\body`, and therefore this 5 vs. 11 is not relevant. The idea behind including this variant here is that the FIFO principle can lead to simple encoding of tail recursion even when no arguments are processed.

Variable number of parameters. TeX macros can take at most 9 parameters. The above `\fifo` macro can be seen as a macro which is relieved from that restriction. Every group, or admissible token, in the input stream after `\fifo` up to and including `\ofif`, will become an argument to the macro.

⁶ So are TeX's `\if`-s after expansion.

⁷ If a list has to be *created*, Knuth's list data-structure might be used, however, simplifying the execution of the list. See *The TeXbook*, Appendix D.2.

When the `\ofif` token is reached, the recursion—that is reading of arguments—will be terminated.⁸

Unknown number of arguments. Tutelaers (1992), as mentioned by Eijkhout (1991), faced the problem of inputting a chess position. The problem is characterized by an unspecified number of positions of pieces, with for the pawn positions the identification of the pawn generally omitted. Let us denote the pieces by the capital letters K(ing), Q(ueen), B(ishop), (k)N(ight), R(ook), and P(awn), with the latter symbol default. The position on the board is indicated by a letter a, b, c, ..., or h, followed by a number, 1, 2, ..., or 8. Then, for example,

```
\position{Ke1, Qd1, Na1, e2, e4}
```

should entail the invocations

```
\piece{K}{e1}\piece{Q}{d1}\piece{N}{a1}
\piece{P}{e2}\piece{P}{e4}
```

This can be done by an appropriate definition of `\position`, and an adaptation of the `\fifo` template, via

```
\def\position#1{\fifo#1,\ofif,}
\def\fifo#1,{\ifx\ofif#1\ofif
  \fi\process#1\relax\fifo}
\def\ofif#1\fifo{\fi}
\def\process#1#2#3{\ifx\relax#3
  \piece{P}{#1#2}\else\piece#1{#2#3}\fi}
```

With the following definition (simplified in relation to Tutelaers)

```
\def\piece#1#2{ #1-#2}
```

we get K-e1 Q-d1 N-a1 P-e2 P-e4.

For an unknown number of arguments at two levels see the Nested FIFO section.

Citation lists. In a list of citations it is a good habit to typeset three or more consecutive numbers as a range. For example 1, 2, 3 as 1–3. This must be done via macros when the numbers are represented by symbolic names, which get their value on the fly. In general the sequence must be sorted⁹ before typesetting. This has been elaborated by Arseneau

⁸ Another way to circumvent the 9 parameters limitation is to associate names to the quantities to be used as arguments, let us say via `def`'s, and to use these quantities via their names in the macro. This is Knuth's parameter mechanism and is functionally related to the so-called keyword parameter mechanism of command languages, and for example ADA.

⁹ The sorting of short sequences within `TEX` has been elaborated by Jeffreys 1990, and myself in Syntactic Sugar.

(1992) in a few `LATEX` styles, and for plain `TEX` by myself. I used the FIFO paradigm in the trivial, stepping-stone, variant of typesetting an explicit non-descending sequence in range notation. The resulting 'process'-macro could be used in the general case, once I realized that FISO—First-In-Smallest-Out—was logically related to FIFO: the required elements are yielded one after the other, whether the first, the last, the smallest, or ... you name it. Perhaps this is a nice exercise for the reader. For a solution see van der Laan (1993).

Vowels, voilà. Schwarz (1987) coined the problem to print vowels in bold face.¹⁰ The problem can be split into two parts. First, the general part of going character by character through a string, and second, decide whether the character at hand is a vowel or not.

For the first part use `\fifo` (or Knuth's `\dolist`).

For the second part, combine the vowels into a string, `aeiou`, and the problem can be reduced to the question $\langle \text{char} \rangle \in \text{aeiou}$? Earlier, I used this approach in searching a card in a bridge hand, van der Laan (1990, the macro `\strip`). That was well-hidden under several piles of cards, I presume? The following encoding is related to `\ismember` (The `TEXbook`, p.379).

```
\newif\iffound
\def\loc#1#2{\%locate #1 in #2
  \def\locate##1##2\end{\ifx\empty##2%
    \empty\foundfalse\else\foundtrue\fi}\%
  \locate#2#1\end}
```

Then `\fifo Audacious\ofif` yields `Audacious`, with

```
\def\process#1{\uppercase{\loc#1}%
  {AEIOU}\iffound{\bf#1}\else#1\fi}
```

Variation. If in the invocation `\locate#2#1` a free symbol is inserted between `#2` and `#1`, then `\loc` can be used to locate substrings.¹¹ And because $\{string_1 \in string_2\} \wedge \{string_2 \in string_1\} \Rightarrow string_1 = string_2$, the variant can be used for the equality test for strings. See also the Multiple FIFO subsection, for general and more effective alternatives for equality tests of strings.

¹⁰ His solution mixes up the picking up of list elements and the process to be applied. Moreover, his nesting of `\if`-s in order to determine whether a character is a vowel or not, is not elegant. Fine (1992)'s solution, via a switch, is not elegant either.

¹¹ Think of finding 'bb' in 'ab' for example, which goes wrong without the extra symbol.

Processing lines. What about processing lines of text? In official, judicial, documents it is a habit to fill out lines of text with dots.¹² This can be solved by making the end-of-line character active, with the function to fill up the line. A general approach where we can \process the line, and not only append to it, can be based upon \fifo.

One can wonder, whether the purpose can't be better attained, while using TeX as formatter, by filling up the last line of paragraphs by dots, because TeX justifies with paragraphs as units.

Processing words. What about handling a list of words? This can be achieved by modifying the \fifo template into a version which picks up words, \fifow, and to give \processw an appropriate function.

```
\def\fifow#1 {\ifx\wofif#1\wofif\fi
  \processw{#1}\fifow}
\def\wofif#1\fifow{\fi}
```

Underlining words. In print it is uncommon to emphasize words by underlining. Generally another font is used; see discussion of exercise 18.26 in *The TeXbook*. However, now and then people ask for (poor man's) underlining of words. The following \processw definition underlines words picked up by \fifow

```
\def\processw#1{\vtop{\hbox{\strut#1
  \hrule}}}
```

Then

```
\leavevmode\fifow leentje leerde lotje
lopen langs de lange lindenlaan \wofif
\unskip.
```

yields leentje leerde lotje lopen langs de lange lindenlaan.

Nested FIFO

One can nest the FIFO paradigm. For processing lines word by word, or words character by character.

Words character by character. Ex11.5 can be solved by processing words character by character. A solution to a slightly simplified version of the exercise reads

```
\fifow Though exercise \wofif \unskip.
%with
\def\processw#1{\fifow#1\ofif}
\def\process#1{\boxit#1}
\def\boxit#1{\setbox0=\hbox{#1}\hbox
  {\lower\dp0\vbox{\offinterlineskip\hrule
```

¹² The problem was posed at EuroTeX '91 by Theo Jurriens.

```
\hbox{\vrule\phantom#1\vrule}\hrule}}}}
yields 
```

In the spirit of \dolist..., ex11.5, is

```
%variant neglecting word structure
\def\fifo{\afterassignment\tap
  \let\nxt=
\def\tap{\ifx\nxt\ofif\ofif
  \fi\process\nxt\fifo}
\def\ofif#1\fifo{\fi}
\def\process#1{\if\space\nxt\
  \else\boxit#1\fi}
\fifo Though exercise\ofif.
```

with the same result 

Mark up natural data. Data for \h(v)align needs & and \cr marks. We can get plain TeX to append a \cr at each (natural) input line (*The TeXbook*, p.249). An extension of this is to get plain TeX to insert \cs-s, column separators, and \rs-s, row separators, and eventually to add \lr, last row, at the end, in natural data. For example prior to an invocation of \halign, one wants to get plain TeX to do the transformation

P*ON \Rightarrow P\cs*\csO\csN\rsD\csE\csK\cs*\lr
DEK*

This can be done via

```
$$\vcenter{\hbox{P*ON}\kern.5ex
  \hbox{DEK*}}\rightarrow,\rightarrow,
%And now right, mark up part
\bdata P*ON
DEK*
\edata\markup\data
\vcenter{\hbox{\data}}$$
with
\def\bdata{\bgroup\obeylines\store}
\def\store#1\edata{\egroup\def\data{#1}}
\def\markup#1{\ea\xdef\ea#1\ea\ea
  \fifol#1\lofif}}
```

and auxiliaries

```
\let\nx=\noexpand
{\catcode`^\M=13
\gdef\fifol#1^\M#2{\fifow#1\ofif%
  \ifx\lofif#2\nx\lr\lofif
  \fi\nx\rs\fifol#2}
\def\lofif#1\lofif{\fi}
\def\fifow#1#2{\fifow#1\ifx\ofif#2\ofif
  \fi\nx\cs\fifow#2}
\def\ofif#1\ofif{\fi}
%with for this example
\def\cs{\sevenrm{\tt\char92}cs}
\def\rs{\sevenrm{\tt\char92}rs}}
```

```
\def\lr{{\sevenrm{\tt\char92}lr}}
```

The above came to mind when typesetting cross-words,¹³ van der Laan (1992b,-d),¹⁴ while striving after the possibility to allow natural input, independent of \halign processing.

Multiple FIFO

What about FIFO for more than one stream?¹⁵ For example comparing strings, either for equality or with respect to lexicographic ordering? Eijkhout (1992, p.137, 138) provided for these applications the macros

```
\ifAllChars... \Are... \TheSame,  
and  
\ifallchars... \are... \before.
```

The encodings are focused at mouth processing. The latter contains many \expandafter-s.

A basic approach is: loop through the strings character by character, and compare the characters until either the assumed condition is no longer true, or the end of either one of the strings, has been reached.

Equality of strings. The TeX-specific encoding, where use has been made of the property of \ifx for control sequences, reads

```
\def\eq#1#2{\def\st{#1}\def\nd{#2}  
\ifx\st\nd\eqtrue\else\eqfalse\fi}
```

with auxiliary \newif\ifeq.

As a stepping stone for lexicographic comparison, consider the general encoding

```
\def\eq#1#2{\continuetrue\eqtrue  
\loop\ifx#1\empty\continuefalse\fi  
\ifx#2\empty\continuefalse\fi  
\ifcontinue \nxte#1\nxtt \nxte#2\nxtu  
\ifx\nxtt\nxtu  
\else\eqfalse\continuefalse\fi  
\repeat  
\ifx\empty#1\ifx\empty#2  
\else\eqfalse\fi\else\eqfalse\fi}
```

with auxiliaries

```
\newif\ifcontinue\newif\ifeq  
\def\nxte#1#2{\def\pop##1##2\pop{  
\gdef#1##2}\gdef#2##1}\ea\pop#1\pop}
```

Then

¹³ With *, or \sqcup , given an appropriate function.

¹⁴ In the latter article I set the puzzles via direct use of nested FIFO. No \halign use nor mark up phase.

¹⁵ For simplicity the streams are stored in def-s, because \read inputs lines.

```
\def\t{abc}\def\u{ab}  
\eq\t\u\ifeq$abc=ab$\else$abc\not=ab$\fi  
yields abc  $\neq$  ab.
```

Lexicographic comparison. Assume that we deal with lower case and upper case letters only. The encoding of \sle—String Less or Equal—follows the same flow as the equality test, \eq, but differs in the test, because of TeX's expansion mechanisms

```
\def\sle#1#2{#1, #2 are def's  
\global\sletrue\global\eqtrue  
\continuetrue  
\loop\ifx#1\empty\continuefalse\fi  
\ifx#2\empty\continuefalse\fi  
\ifcontinue \nxte#1\nxtt \nxte#2\nxtu  
\ea\ea\ea\lle\ea\nxtt\nxtu  
\repeat  
\ifeq\ifx\empty#2\ifx\empty#1  
\else\global\slefalse\fi\fi  
\fi}
```

with auxiliaries (\lle=Letter Less or Equal)

```
\newif\ifcontinue  
\global\newif\ifsle\global\newif\ifeq  
\def\nxte#1#2{\def\pop##1##2\pop{  
\xdef#1##2}\xdef#2##1}\ea\pop#1\pop}  
\def\nxe#1#2{\uppercase{\ifnum'#1='#2}  
\else\continuefalse\global\eqfalse  
\uppercase{\ifnum'#1>'#2}\global  
\slefalse\fi  
\fi}
```

For example

```
\def\t{ABC}\def\u{ab}\sle\t\u  
\ifsle$ABC\le ab$\else$ABC>ab$\fi  
yields ABC > ab;
```

```
\def\t{aa}\def\u{a}\sle\t\u  
\ifsle$aa\le a$\else$aa>a$\fi  
yields aa > a;
```

```
\def\t{aa}\def\u{b}\sle\t\u  
\ifsle$aa\le b$\else$aa>b$\fi  
yields aa  $\le$  b;
```

```
\def\t{noo}\def\u{open}\sle\t\u  
\ifsle$noo<open\else$noo>open$\fi  
yields noo > open.
```

The above can be elaborated with respect to \read for strings each on a separate file, to strings with accented letters, to the inclusion of an ordering table, and in general to sorting. Some of the mentioned items will be treated in Sorting in BLUe, to come.

LIFO

A modification of the `\fifo` macro—`\process{#1}` invoked at the end instead of at the beginning—will yield the Last-In-First-Out template. Of course LIFO can be applied to reversion on the fly, without explicitly allocating auxiliary storage.¹⁶

```
\def\lifo#1#2\ofil{\ifx\empty#2
  \empty\ofil\fi\lifo#2\ofil\process#1}
\def\ofil#1\ofil{\fi}
```

The test for emptiness of the second argument is similar to the `\TeXnique` used by Knuth in `\displaytest` (*The `\TeX`book*, p.376): `\if!#3!....`

With the identity—`\def\process#1{#1}`, or the invoke `\process#1` replaced by `#1`¹⁷—the template can be used for reversion on the fly. For example `\lifo aap\ofil` yields `paa`.¹⁸

Change of radix. In *The `\TeX`book* a LIFO exercise is provided at p.219: print the digits of a number in radix 16 representation. The encoding is based upon the property

$$(N \div r^k) \bmod r = d_k, \quad k = 0, 1, \dots, n,$$

with radix `r`, coefficients `dk`, and the number representation

$$N = \sum_{k=0}^n d_k r^k.$$

There are two ways of generating the numbers `dk`: starting with `dn`, or the simpler one starting with `d0`, with the disadvantage that the numbers are generated in reverse order with respect to printing. The latter approach is given in *The `\TeX`book*, p.219. Adaptation of the LIFO template does not provide a solution much different from Knuth's, because

¹⁶ Johannes Braams drew my attention to Knuth and MacKay (1987), which contained among others `\reflect... \tcelfer`. They compare `#1` with `\empty`, which is nice. The invocation needs an extra token, `\empty`—a so-called sentinel, see Wirth (1976)—to be included before `\tcelfer`, however. (Knuth and Mackay hide this by another macro which invokes `\reflect... \empty \tcelfer`). My approach requires at least one argument, with the consequence that the empty case must be treated separately, or a sentinel must be appended after all.

¹⁷ Remember the stack size limitations.

¹⁸ Note that Knuth's test `\if!#3!....` goes wrong for `#3` equals `!`, and similarly my use of the idea goes wrong for `#2` equals `\empty`, which is not 'empty'. Given the context those situations don't occur, however.

the numbers to be typeset are generated in the recursion and not available in the input stream.

Acknowledgements

Włodek Bzyl and Nelson Beebe are kindly acknowledged for their help in clearing up the contents and correcting my use of English, respectively.

Conclusion

In looking for a fundamental approach to process elements sequentially—not to confuse with list processing where the list is also built up, see *The `\TeX`book*, Appendix D.2, or with processing of *every* token in the input stream, see ex11.5 or p.376—`\TeX` templates for FIFO and LIFO emerged.

The templates can be used for processing lines, words or characters. Also processing of words line by line, or characters word by word, can be handled via nested use of the FIFO principle.

The FIFO principle along with the look ahead mechanism is applied to molding natural data into representations required by subsequent `\TeX` processing.

Courseware might benefit from the FIFO approach to unify answers of the exercises of the macro chapter.

`\TeX`'s `\ifx...` and `\if...` conditionals are non-commutative with respect to their *first level* operands, while the similar mathematical operations are, as are the operations in current high-level programming languages.

Multiple FIFO, by comparing strings lexicographically, has been touched upon.

References

- [1] Arseneau, D. (1992): `overcite.sty`, `drftcite.sty`, `cite.sty`. From the file server.
- [2] Eijkhout, V. (1991): `\TeX` by Topic. Addison-Wesley.
- [3] Fine, J. (1992): Some basic control macros for `\TeX`. *TUGboat* 13, no. 1, 75–83.
- [4] Hendrickson, A. (priv. comm.)
- [5] Jeffrey, A (1990): Lists in `\TeX`'s mouth. *TUGboat* 11, no. 2, 237–244.
- [6] Kabelschat, A. (1987): `\expandafter` vs. `\let` and `\def` in conditionals and a generalization of plain's `\loop`. *TUGboat* 8, no. 2, 184–185.
- [7] Knuth, D.E. (1968): The Art of Computer Programming. 1. Fundamental Algorithms. Addison-Wesley.

- [8] Knuth, D.E. (1984): The *TeXbook*. Addison-Wesley.
- [9] Knuth, D.E., P. Mackay (1987): Mixing right-to-left texts with left-to-right texts. *TUGboat* 7, no. 1, 14–25.
- [10] Laan, C.G. van der (1990): Typesetting Bridge via *TeX*, *TUGboat* 11, no. 2, 91–94. Also MAPS91.2, 51–62.
- [11] Laan, C.G. van der (1992a): Tower of Hanoi, revisited. *TUGboat* 13, no. 1, 91–94. Also MAPS92.1, 125–127.
- [12] Laan, C.G. van der (1992b): Typesetting Crosswords via *TeX*. EuroTEX '92, 217–224. Also MAPS92.1, 128–131.
- [13] Laan, C.G. van der (1992c): Table Diversions. EuroTEX '92, 191–211. Also a little adapted in MAPS92.2, 115–128.
- [14] Laan, C.G. van der (1992d): Typesetting Crosswords via *TeX*, revisited. MAPS92.2, 145–146.
- [15] Laan, C.G. van der (1992e): Syntactic Sugar. MAPS92.2, 130–136. (Submitted TUG'93.)
- [16] Laan, C.G. van der (1993): Typesetting number sequences. MAPS93.1, 4p.
- [17] Laan, C.G. van der (in progress): Sorting in BLUE. MAPS93.1. (Submitted TUG'93. For heap sort encoding in plain *TeX*, see MAPS92.2, 137–138.)
- [18] Salomon, D. (1992): Advanced *TeX* course: Insights & Hindsights, MAPS 92 Special. 254p.
- [19] Schwarz, N. (1987): Einführung in *TeX*, Addison-Wesley.
- [20] Tutelaers, P. (1992): A font and a style for typesetting chess using *LATeX* or *TeX*. *TUGboat* 13, no. 1, 85–90.
- [21] Wirth, N. (1976): Algorithms + Data Structures = Programs. Prentice-Hall.

◊ Kees van der Laan
 Hunzeweg 57, 9893PB
 Garnwerd (Gr), The Netherlands
 cgl@rug.nl



An Update on the **babel** System

Johannes Braams

Abstract

This article describes the changes that have been made to the **babel** system since the article describing the system appeared in *TUGboat* 12, no. 2. This article announces the release of a new version of the **babel** system.

1 Introduction

Since the publication of the **babel** system in *TUGboat* [1] several changes have occurred. With the new release of *LATeX* — which appeared at the end of 1991 — the internationalised version *ILATeX*, prepared by Joachim Schrod [2], was withdrawn. But some of its functionality was still needed, so a modification of the **babel** system was necessary.

Besides this a couple of bugs were reported and had to be fixed. The major problem was that the language changing commands were not ‘local’, they contained global definitions. In the current version these commands obey grouping correctly.

Some macros that formerly were in language-specific files have been moved to the core of the system, because they are being used in several language-specific files.

2 Changes to the core of **babel**

The changes to the core of the **babel** system are the most extensive.

\selectlanguage

The **babel** user-command `\selectlanguage` now also accepts a control sequence as its argument. This was included to provide compatibility for users who were used to the syntax of the original `german.tex`, but wanted to switch to **babel**. The escape character is ‘peeled off’ and the name of the control sequence is then used as the name of the language to select.

Another change to the `\selectlanguage` macro is that it now stores the name of the current language in the control sequence `\languagename`. The

contents of this control sequence could be tested in the following way:

```
\edef\tmp{\string english}
\ifx\languagename\tmp
...
\else
...
\fi
```

The construction with `\string` is necessary because `\languagename` returns the name with characters of category code 12 (other).

Saving macro definitions

A new way of handling macros that are temporarily redefined was developed by Bernd Raichle and included in the core of the `babel` system. Two new macros for use in the language-specific files have been introduced.

These macros, `\babel@savevariable<register>` and `\babel@save<macro>`, append code to `\originalTeX`. This code restores the value (or meaning) of what was saved when `\originalTeX` was executed.

Special characters

Some of the language-specific files introduce one or more characters that are special in some way. Such characters have to be added to `\dospecials` (and `\@sanitize` too for `LATEX`) whenever their special meaning is activated. But they may have to be removed again when another language, which doesn't use them, is in effect.

To this end two new control sequences, that are meant to be used in the language-specific files, are introduced. They are `\babel@add@special` and `\babel@remove@special` and perform the necessary tasks.

Additional facilities

A specific request from Joachim Schrod for `babel` was the possibility to extend the definition of a control sequence on the fly. It should, for instance, be possible that the user adds a macro of his own to the definition of `\extrasenglish`.

This feature is now provided by the macro `\addto{<control sequence>}{{<TEX code>}}`. It is now used throughout the language-specific files to build the macros `\extras{lang}` and `\noextras{lang}`.

The support macros `\allowhyphens`, `\set@low@box` and `\save@sf@q` have been moved from the language-specific files to the core of the `babel` system.

2.1 The files

In the previous release a file called `latexhax.com` was provided. This was needed to provide some macros normally defined by `LATEX`, to plain `TeX` users. The need for this file has been removed in the current release of the `babel` system.

In the previous release of the system, four different files were provided (all derived from `hyphen.doc`) that were needed for different combinations of versions of `TeX` and `plain.tex` or `lplain.tex`. This has been changed. In the current version only two different files are derived from `hyphen.doc`. They are `babel.switch` and `babel.hyphen`. The file `babel.switch` is needed for people who can't build a new format or don't have `TeX` version 3. The file `babel.hyphen` should be loaded into the format by `iniTeX`. It provides the macros from `babel.switch`, but additionally it reads the file `language.dat`, which specifies the languages for which hyphenation patterns should be loaded.

In the previous release the file `babel.com` contained redefinitions for a lot of `LATEX` macros to replace texts with control sequences. This has been removed, because it is no longer necessary for releases of `LATEX` dated December 1991 or later. Those who still have an older release of `LATEX` can produce a special version of `babel.com` by including the `docstrip` option `<names>` when stripping the file `babel.doc`.

With the release of the new version of Frank Mittelbach's `doc` package the stripped files are no longer distributed. The `babel` distribution now includes a file `install.babel` with which you can produce them (give the command `tex install.babel`).

3 Changes to the language specific files

Bernd Raichle has invented a solution for things like `\char"45` when the `"` is active. His solution (from `german 2.3e`) has been included in `germanb` and is copied for other language specific files that have an active `"`.

A few terms have been added to the `\captions{...}` macros, again following `german.tex`. These terms are `\prefacename`, `\seename` and `\seealso`. I don't have the correct translations

for all languages yet, but that will be repaired as soon as someone provides them to me.

For the Dutch language the behaviour of the active double quote has been slightly modified. It has been noted that there is a difference between "e, where a 'trema' should be produced and \"u, where we should get an 'umlaut'.¹ The difference between the two is that the 'trema' should disappear at a hyphenation point, whereas the 'umlaut' should not.

References

- [1] Johannes Braams, *Babel, a multilingual style-option system for use with LATEX's standard document styles*, TUGboat 12 (1991), no. 2, pp. 291–301.
- [2] Joachim Schrod, *International LATEX is ready to use*, TUGboat 11 (1990), no. 1, pp. 87–90.

◊ Johannes Braams
 PTT Research Neher Laboratories
 P.O. Box 421
 2260 AK Leidschendam
 The Netherlands
 J.L.Braams@research.ptt.nl

Hacker's Guide to *AMS*Fonts and NFSS in the Context of LATEX

Rafał Źbikowski

Abstract

The purpose of this document is to describe briefly *AMS*Fonts and the New Font Selection Scheme (NFSS) in the context of LATEX. The issues addressed are as follows.

***AMS*Fonts:** What are *AMS*Fonts? Where to get *AMS*Fonts from? How to install *AMS*Fonts for LATEX?

New Font Selection Scheme: What is the New Font Selection Scheme (NFSS)? Why to use NFSS? Where to get NFSS from? How to install NFSS? How to use NFSS to install *AMS*Fonts for LATEX?

Also: How can NFSS and *AMS*Fonts be used in practice? (Examples.)

An attempt is made to answer these questions from the user's point of view as opposed to a (I)TEXpert's/designer's.

— * —

1 *AMS*Fonts

This section explains what *AMS*Fonts are, where to get them from and how to install them.

1.1 What are *AMS*Fonts?

*AMS*Fonts¹ is an additional set of fonts (absent in distributions of TEX and LATEX). The most recent version, released in August 1991, is known as *AMS*-Fonts Version 2.1.² *AMS*Fonts contains over two hundred mathematical symbols (like \leq , \emptyset , $\dot{+}$, $\ddot{\cdot}$, \circlearrowleft , etc.) and also so-called Euler fonts, e.g. \mathfrak{E} , E , \mathcal{E} . It also has a special alphabet (Blackboard bold) with \mathbb{R} for the real numbers, \mathbb{C} for complex numbers and so on. Finally, the Russian alphabet (including pre-1917 characters like \mathbf{e}), or cyrillic, is available plus letters needed for Ukrainian, Serbian and Bulgarian.

It should be emphasised that, except for cyrillic, which is a text font, *AMS*Fonts are designed to extend the available range of symbols and alphabets for *mathematics*.

¹ *AMS* stands, obviously, for the American Mathematical Society.

² From now on, when talking about *AMS*Fonts, this will mean *AMS*Fonts Version 2.1.

¹ Editor's note: 'Trema' (English 'diaeresis') is the " mark placed over a vowel to indicate its pronunciation in a separate syllable; 'umlaut' indicates a vowel that has undergone linguistic modification.

1.2 Where to get *AMS*Fonts from?

The original distribution site for *AMS*Fonts is
e-math.ams.org (130.44.1.100)

and the directory is

/ams

available via ftp.

Users having UNIX-compatible compress (uncompress) and tar (untar) utilities (versions also exist for DOS and VMS) can get the following (binary) files³

```
632033 amsfonts-sources.tar.Z
76443 tfm-files.tar.Z
2449408 amsfonts118.tar
3235840 amsfonts180.tar
3784704 amsfonts240.tar
4907008
amsfonts300.tar
6512640 amsfonts400.tar
```

from the ams directory, which covers the whole distribution together with documentation printable with plain TeX. The files amsfonts\$\$.tar (where \$\$ is 118 or 180 or 240 or 300 or 400) contain .pk files, the number \$\$ indicating the required printer/previewer resolution in dots per inch (dpi).

Note that amsfonts\$\$.tar are not compressed using UNIX's compress facility.

Users not having the UNIX-compatible utilities will have to pull the files from subdirectories

```
/ams/amsfonts
/ams/amsfonts/doc
/ams/amsfonts/pk-files
/ams/amsfonts/sources
/ams/amsfonts/sources/cyrillic
/ams/amsfonts/sources/euler
/ams/amsfonts/sources/extracm
/ams/amsfonts/sources/symbols
/ams/tfm-files
```

Subdirectory ams/amsfonts/pk-files contains .pk files, organised in directories according to the required printer (previewer) resolution, i.e.

```
/ams/amsfonts/pk-files/118dpi
/ams/amsfonts/pk-files/180dpi
/ams/amsfonts/pk-files/240dpi
/ams/amsfonts/pk-files/300dpi
/ams/amsfonts/pk-files/400dpi
```

Files of the *AMS*Fonts distribution are rather big, even in the compressed form (as seen from the above listings). It is recommended to pull only the relevant files (especially the .pk ones). For example, files necessary for a 300dpi installation requiring

632033 amsfonts-sources.tar.Z

4907008 amsfonts300.tar

76443 tfm-files.tar.Z

occupy ca. eight megabytes in uncompressed form.

1.2.1 Documentation

Documentation (the *AMS*Fonts Version 2.1 User's Guide) can be found in

/ams/amsfonts/doc

under the name

userdoc.tex

To compile it you need to have the following files

```
amssym.def
amssym.tex
cyracc.def
userdoc.cyr
userdoc.def
userdoc.ins
userdoc.fnt
```

and also .tfm (from tfm-files.tar.Z) and .pk files (from amsfonts300.tar or whatever resolution is appropriate). The subdirectory amsfonts/doc contains by default all userdoc.* files but you can find the first three (i.e. amssym.def, amssym.tex, cyracc.def) in the directory amsfonts. Once all files are gathered, type

prompt> tex userdoc

This should compile smoothly and produce userdoc.dvi (41 pages). Provided you put *AMS*Fonts' .pk files in the place where your previewer (printer) looks for it, you should be able to see (print) it.

Read userdoc or at least have a glance at the provided font tables to get an idea what you can expect from it.

1.3 How to install *AMS*Fonts for LATEX?

It is assumed here that you already have all the files of *AMS*Fonts. Also, you should have a copy of the *AMS*Fonts User's Guide printed out.

The User's Guide says almost nothing about installation of *AMS*Fonts for LATEX (see the bottom of page 11), but you should have a copy of it for reference. It gives the command names of additional math symbols, among others.

To use *AMS*Fonts smoothly under LATEX you need to get and install the New Font Selection Scheme (NFSS). This is described below.

³ Numbers indicate sizes (in bytes) of the files.

2 New Font Selection Scheme (NFSS) for L^AT_EX

This section explains what the New Font Selection Scheme is, why to use it, how to install it and, finally, how to use it together with *AMS*Fonts.

2.1 What is the New Font Selection Scheme (NFSS)?

The New Font Selection Scheme (NFSS) is a new version of *lfonts.tex* file written by L^AT_EXperts Frank Mittelbach and Rainer Schöpf. When an old *lfonts.tex* is replaced by the new one and you recompile your L^AT_EX with the *new lfonts.tex*, you have at your disposal all the commands and properties of NFSS. These allow you to load any nonstandard (and standard, i.e. those coming with a distribution of T_EX and L^AT_EX) L^AT_EX fonts on demand (i.e. when you really want them without memory-consuming preloading) *both* in text and math mode. It is much better than the standard L^AT_EX solution (see p. 116 and p. 200 of Leslie Lamport's *L^AT_EX User's Guide*, Addison-Wesley, Reading, Mass., 1985, ISBN 0-201-15790-X).

Thus, the name NFSS means a set of rules for loading fonts that are available to a user who replaced his/her old *lfonts.tex* L^AT_EX file with the new one coming with the distribution of NFSS. Also, NFSS has a backward compatibility option. In actual fact *lfonts.new* consists of NFSS + L^AT_EX adaptions. NFSS is by no means restricted to L^AT_EX; it works equally well with plain T_EX, but needs another set of interface macros. This will not be addressed here, since this document deals with L^AT_EX only.

NFSS is a serious enhancement of L^AT_EX offering a swift, simple and uniform method for using nonstandard (and standard) L^AT_EX fonts. It is the *only* practicable method of using *AMS*Fonts and PostScript fonts with L^AT_EX. When installed it also allows you to use the old font selection scheme.

2.2 Where to get NFSS from?

The original distributing ftp site⁴ for NFSS is

129.69.1.12

Note that the node has two *equivalent* names: either
ftp.uni-stuttgart.de (129.69.1.12)
or

⁴ The archive at Stuttgart is accessible by electronic mail as well, under the address *mail-server@rus.uni-stuttgart.de*

rusmv1.rus.uni-stuttgart.de (129.69.1.12)
and the directory is

/pub/soft/tex/macros/latex/styles/base/NFSS
Communication with this machine may be not too fast, so be patient.

Directory

/pub/soft/tex/macros/latex/styles/base/NFSS
contains the following (ASCII text) files

SUBDIRECTORY	12718 array.sty 4027 basefont.tex 11888 concrete.doc 10760 dclfont.sty 8951 euscript.doc 8172 exscale.doc 22563 fontdef.dc 27992 fontdef.max 15338 fontdef.ori 6405 install.mz3 36907 lfonts.new 2837 margid.sty 4831 newlfont.sty
SUBDIRECTORY	{\NFSS}-addons 12881 {\NFSS}.bug 40893 {\NFSS}.tex 10224 {\NFSS}inst.tex 9442 {\NFSS}prob.tex 2869 nomargid.sty 4989 oldlfont.sty 4692 preload.dc 4570 preload.min 4646 preload.ori 4125 preload.xpt
SUBDIRECTORY	ps{\NFSS} 5381 readme.mz 3993 readme.mz3 5550 scripts.doc 4399 syntonly.sty 6650 tracefnt.sty

The file to replace *lfonts.tex* is *lfonts.new* which loads *fontdef.tex* and one of *preload.**⁵. The files in subdirectories *NFSS-addons* and *psNFSS*

⁵ To install L^AT_EX, IniT_EX should be run. When IniT_EX is run with *lplain.tex* as the input file a point is reached when T_EX wants to read in *lfonts.tex*. Here *lfonts.new* should be specified instead. At some point in processing *lfonts.new* IniT_EX will ask for *xxxlfont.sty*, which does not exist. The appearance of the name *xxxlfont.sty* in the source of *lfonts.new* is a convenient stop to allow the choosing one of

do not belong to the proper distribution of NFSS and will not be described here.

2.2.1 Documentation of NFSS

Documentation of the New Font Selection Scheme (NFSS) is composed of three parts: *NFSS.tex*, a copy of the original article, published in *TUGboat*, by Mittelbach and Schöpf; *NFSSinst.tex*, installation guide; *NFSSprob.tex*, possible problems (and fixes) that may occur during installation.

These are ordinary L^AT_EX files (requiring a L^AT_EX version not older than the Dec. 91 release⁶), but *NFSS.tex* makes use of *array.sty* (provided with the NFSS distribution) and *twocolumn.sty* (provided with recent L^AT_EX distributions; note the name of the file: without 'n'). You should be able to print out the documentation (see below), but read *readme.mz3* first.

To compile *NFSS.tex* many varieties of Computer Modern Sans Serif font are required, so it is advisable to have the appropriate *.tfm* and *.pk* files ready before typing:

```
prompt> latex {\NFSS}
```

It will report errors unless a recent version of L^AT_EX is used or when sans serif fonts are missing. The first type of error is fatal; the second can be overcome by pressing <return> enough times. However, the output will be poor and may lead to misinformation (slanted, etc., shapes are used in important examples). If the compilation was error-free, the file can be previewed/printed. Then type

```
prompt> latex {\NFSS}inst
```

```
prompt> latex {\NFSS}prob
```

These should compile smoothly, since the files use Computer Modern Roman only.

Installation is described in detail in the file *NFSSinst.tex*, and in case of problems consult *NFSSprob.tex*.

the three options: *oldfont.sty*, *newfont.sty*, *basefont.tex*. For details read *NFSSinst.tex*.

⁶ The most recent version <25 March 1992> is available from archive in Stuttgart (129.69.1.12) from directory /pub/soft/tex/macros/latex; also archive Niord.SHSU.edu offers it via ftp (read [.LATEX]0000README.FTP_USERS) or by mail (send a message to FILESERV@SHSU.edu with the body SENDME LaTeX).

2.2.2 *AMS-L^AT_EX*

It is not essential, but helpful (especially for future use), to get a copy of the *AMS-L^AT_EX* manual, whose Part II (pages 4–17) describes in detail the usage and principles of NFSS. It also gives valuable clues about using *AMSF*onts via NFSS. To get this ftp to

```
e-math.ams.org (130.44.1.100)
```

and get the (binary) file

```
588389 amslatex.tar.Z
```

from directory

```
/ams
```

or pull files from directory

```
/ams/amslatex
```

and its subdirectories

```
/ams/amslatex/doc  
/ams/amslatex/fontsel  
/ams/amslatex/inputs  
/ams/amslatex/latex
```

These will give the *whole* distribution of *AMS-L^AT_EX*, which is not needed to print out the *AMS-L^AT_EX* manual. To get this go to subdirectory /ams/amslatex/doc to find file *amslatex.tex*. This document can be processed using *ordinary L^AT_EX*. To generate it type

```
prompt> latex amslatex
```

Everything should go smoothly resulting in an *amslatex.dvi* file (69 pages long).

AMS-L^AT_EX includes by default NFSS (see files in /ams/amslatex/fontsel), so you may want to install *AMS-L^AT_EX* altogether and use it for the purposes of NFSS. If you don't, retain the following .sty files (for use under ordinary L^AT_EX)

```
amsbsy.sty  
amsfonts.sty  
amssymb.sty
```

which can be found in

```
/ams/amslatex/inputs
```

and are very handy for swift use of *AMSF*onts for L^AT_EX.

2.3 How to install NFSS?

The best answer to this question is contained in the file *NFSSinst.tex*, which comes with the distribution of NFSS. See also footnote 5.

3 How can NFSS and $\mathcal{A}\mathcal{M}\mathcal{F}\text{onts}$ be used in practice?

Here several examples are provided grouped in two sections. To compile them NFSS was installed together with $\mathcal{A}\mathcal{M}\mathcal{F}\text{onts}$. The `fontdef` option, chosen when installing NFSS, was `fontdef.max`.

3.1 $\mathcal{A}\mathcal{M}\mathcal{F}\text{onts}$ and NFSS in Math Mode

This section shows examples of defining fonts for use in math mode using NFSS. The fonts employed are those provided with the package $\mathcal{A}\mathcal{M}\mathcal{F}\text{onts}$. These are

- Euler fraktur,
- Euler roman,
- Euler script,
- University of Washington cyrillic,

and also Blackboard bold. It is shown here how to use all the fonts both in normal and bold versions (except for Blackboard bold, which doesn't have a bold version).

The following commands are defined

- `\eufm` for Euler fraktur medium (as opposed to bold),
- `\eurm` for Euler roman medium,
- `\eusm` for Euler script medium,
- `\cyr` for University of Washington cyrillic medium.

There is also a predefined one: `\Bbb` (see **Example 4**). Their effective definitions are shown at the end of this section. To make use of the above-mentioned fonts the following style files should be loaded

```
amsbsy.sty
amstext.sty
cyracc.def
```

This document also makes use of `amssymb.sty` and `amsfonts.sty` to take full advantage of the extended math symbols set provided by $\mathcal{A}\mathcal{M}\mathcal{F}\text{onts}$. The use of an extra symbol is marked by \checkmark , itself a (non-mathematical) symbol from $\mathcal{A}\mathcal{M}\mathcal{F}\text{onts}$.

3.1.1 Examples

This section shows four simple examples of usage of the commands in *math mode*.

Example 1

[Here Euler script is used for the capital O, through a `\eusm` command, and the symbol for the empty set `\varnothing` is used from the extra math symbols B font (file `msbm`).]

DEFINITION. Let X be a non-empty set. Then the pair (X, \mathcal{O}) is called a *topological space* iff

1. X is open,
2. $\emptyset \checkmark$ is open,
3. \mathcal{O} is an open family of subsets of X , i.e.
 - (a) $\forall \mathcal{O}_i \in \mathcal{O}$ the intersection of a finite number of members of \mathcal{O} , i.e. $\bigcap_{i=1}^n \mathcal{O}_i$, is open,
 - (b) $\forall \mathcal{O}_i \in \mathcal{O}$ the union (finite or infinite) of members of \mathcal{O} , i.e. $\bigcup_{i=1}^{\infty} \mathcal{O}_i$, is open.

The family \mathcal{O} is called a *topology on X*. \square

Example 2

[Here Euler fraktur `\eufm` is used for the capital A, Euler roman `\eurm` for the capital J; the symbol of (non-strict) precedence `\preccurlyeq` comes from the extra math symbols A font (file `msam`).]

PROPOSITION. Let $(A, \preccurlyeq) \checkmark$ be a well-ordered set. Then the family \mathfrak{A} of all initial segments of A , i.e. $\mathfrak{A} = \{J \subset A \mid \forall x, y \in A ((y \in J) \wedge (x \preccurlyeq y)) \Rightarrow (x \in J)\}$, together with the relation \subseteq is also a well-ordered set. \square

Example 3

[Here University of Washington cyrillic `\cyr` is used (for the Russian 'Sh' letter) and the solid Halmos' symbol `\blacksquare` comes from the extra math symbols A font (file `msam`).]

NOTATION. We shall denote by III the Shafarevich group and we shall use m_1, \dots, m_n for its subgroups. $\blacksquare \checkmark$

Example 4

[Here the use of the bold mode of Euler roman `\eurm` is shown via the `\boldsymbol` command, e.g. `\boldsymbol{\eurm{x}}` to get x . Also the Blackboard bold font `\Bbb`, defined in `amsfonts.sty`, is used to denote the set of real numbers. Finally, two extra symbols are displayed: `\blacktriangleright` and `\bigstar`, both from the extra math symbols A font (file `msam`).]

► \checkmark **EXERCISE.** Let $f(x, y) = 0$ be given with $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$. State a sufficient condition for the existence of $g: \mathbb{R}^n \rightarrow \mathbb{R}^m$, such that $y = g(x)$ (The Implicit Function Theorem). $\star \checkmark$

3.1.2 Commands' definitions

The command `\Bbb` is defined in the file `amsfonts.sty`. Also a counterpart of `\eufm` is predefined in `amsfonts.sty` as `\frak`.

The verbatim definitions used in this document look as follows.

```
% The following commands should produce
% proper results. To understand how to
```

```
% type Russian when using cyracc.def see
% the {\AmS}Fonts user's guide
% (section 'Cyrillic input', pp. 15--16).

\newmathalphabet{\eufm}
\addtoversion{normal} % Euler
  {\eufm}{euf}{m}{n} % fraktur.
\addtoversion{bold} % Euler
  {\eufm}{euf}{b}{n} % fraktur bold.
% Define command \eufm as Euler fraktur
% font to be used in math mode.
% It is already defined in
% amsfonts.sty as \frak.

\newmathalphabet{\eurm}
\addtoversion{normal} % Euler
  {\eurm}{eur}{m}{n} % roman.
\addtoversion{bold} % Euler
  {\eurm}{eur}{b}{n} % roman bold.
% Define command \eurm as Euler roman
% font to be used in math mode.

\newmathalphabet{\eusm}
\addtoversion{normal} % Euler
  {\eusm}{eus}{m}{n} % script.
\addtoversion{bold} % Euler
  {\eusm}{eus}{b}{n} % script bold.
% Define command \eusm as Euler script
% font to be used in math mode.

\input{cyracc.def} % This file is needed
                  % for cyrillic fonts.

\newmathalphabet{\cyr}
\addtoversion{normal} % UW
  {\cyr}{UWCyr}{m}{n} % cyrillic
\addtoversion{bold} % UW
  {\cyr}{UWCyr}{b}{n} % cyrillic bold
% Define command \cyr as
% University of Washington (UW) Cyrillic
% to be used in math mode.

% To get bold in math use command
% \boldsymbol{} provided by amsbsy.sty
% file. See examples in text.
```

3.2 Examples of application of NFSS to Text Mode

This section shows examples of defining fonts for use in *text mode* using NFSS. The fonts employed are

those provided with the *\AmS*Fonts package.⁷ These are

- Euler fraktur,
- Euler roman,
- Euler script,
- Blackboard bold,
- University of Washington cyrillic.

It is shown here how to use all (but Blackboard bold) the fonts in both normal and bold versions.

The following commands are defined for Euler fonts

- \teufm for Euler fraktur normal,
- \teufb for Euler fraktur bold,
- \teurm for Euler roman normal,
- \teurb for Euler roman bold,
- \teusm for Euler script normal,
- \teusb for Euler script bold,

and also \tBbb for Blackboard bold. Also a set of commands is introduced for University of Washington cyrillic

- \tcyrm for Univ. of Washington cyrillic normal,
- \tcyrb for Univ. of Washington cyrillic bold,
- \tcyrit for Univ. of Washington cyrillic italic,
- \tcyrsc for Univ. of Washington cyrillic small caps.

Their effective definitions are shown at the end of this document. To make use of the cyrillic fonts the file *cyracc.def* is needed.

3.2.1 Examples of Euler fonts and Blackboard Bold

This section shows simple examples of usage of Euler fonts and Blackboard bold in text mode.

The string 'JOHN SMITH is my name.' (with a dot at the end) will be generated using the relevant commands defined above.

1. Euler fraktur normal; command \teufm
 JOHN SMITH is my name.
2. Euler fraktur bold; command \teufb
 JOHN SMITH is my name.
3. Euler roman normal; command \teurm
 JOHN SMITH is my name

⁷ It should be emphasised that *\AmS*Fonts, other than cyrillic, are meant for *math mode* only, i.e. they do *not* have punctuation, numbers, ligatures, etc. However, *any* font intended for text use will give nice results with NFSS. The examples here serve as a 'template' for user-defined font-loading commands and *\AmS*Fonts as an illustration of fonts (mis)use.

4. Euler roman bold; command \teurb
JOHN SMITH is my name
5. Euler script normal; command \teusm
JOHN SMITH \
6. Euler script bold; command \teusb
JOHN SMITH \
7. Blackboard bold; command \tBbb
JOHN SMITH $\beth_1 > \beth_2 \times \beth_3$

The reason for this strange output will become clear after looking at the font tables provided by *User's Guide to AMSFonts Version 2.1* on pp. 35–37.

3.2.2 Examples of University of Washington (UW) Cyrillic

This section shows simple examples of usage of University of Washington cyrillic fonts in text mode.

The string ‘Mikhail Gorbachëv.’ (with a dot at the end) will be generated using the relevant commands defined above

1. UW cyrillic normal; command \tcyrм
Михаил Горбачёв.
Alternatively:
\tcyr Mikhail Gorbach"ev.)
2. UW cyrillic bold; command \tcyrb
Михаил Горбачёв.
Alternatively:
\renewcommand{\bfdefault}{b}
\bf\tcyr Mikhail Gorbach"ev.)
3. UW cyrillic italic; command \tcyrit
Михаил Горбачёв.
Alternatively:
\it\tcyr Mikhail Gorbach"ev.)
4. UW cyrillic small caps; command \tcyrsc
МИХАИЛ ГОРБАЧЁВ.
Alternatively:
\sc\tcyr Mikhail Gorbach"ev.)

To obtain the special cyrillic characters ё, є, і, ѫ, Ѫ, Ѯ, Ѱ, Ѳ, Ѵ, Ѷ, Ѹ, ѻ, Ѽ the command \cyracc, defined in *cyracc.def* is needed (see *User's Guide to AMSFonts Version 2.1* pp. 14–16). The best strategy is to include \cyracc in the definitions of the cyrillic font commands, as shown below in the definitions of \tcyr, \tcyrм, \tcyrb, \tcyrit, and \tcyrsb.

3.2.3 Commands' definitions

The commands involving University of Washington cyrillic rely heavily on the file *cyracc.def*. Note that \cyracc command was added at the end of each definition to allow smooth use of the accented characters of the cyrillic font. The addition is relevant to this *AMS* font *only* and is not necessary

for any others. This means that in any other case a definition should terminate with \selectfont.

```
% The following commands
% (with the exception of Cyrillic fonts)
% can produce imperfect output due to
% the lack of punctuation, numbers,
% ligatures etc. in the source files
% defining them.
```

```
%%% Euler fonts in text mode. %%%
```

```
\newcommand{\teufm}{\fontfamily{euf}%
  \fontseries{m}\fontshape{n}\selectfont}
% Define command \teufm as Euler fraktur
% font to be used in text mode.
```

```
\newcommand{\teufb}{\fontfamily{euf}%
  \fontseries{b}\fontshape{n}\selectfont}
% Define command \teufb as bold Euler
% fraktur font to be used in text mode.
% This can also be achieved by
% typing \bf\teufm.
```

```
\newcommand{\teurm}{\fontfamily{eur}%
  \fontseries{m}\fontshape{n}\selectfont}
% Define command \teurm as Euler roman
% font to be used in text mode.
```

```
\newcommand{\teurb}{\fontfamily{eur}%
  \fontseries{b}\fontshape{n}\selectfont}
% Define command \teurb as bold Euler
% roman font to be used in text mode.
% This can also be achieved by
% typing \bf\teurm.
```

```
\newcommand{\teusm}{\fontfamily{eus}%
  \fontseries{m}\fontshape{n}\selectfont}
% Define command \teusm as Euler script
% font to be used in text mode.
```

```
\newcommand{\teusb}{\fontfamily{eus}%
  \fontseries{b}\fontshape{n}\selectfont}
% Define command \teusb as bold Euler
% script font to be used in text mode.
% This can also be achieved by
% typing \bf\teusm.
```

```
\newcommand{\tBbb}{\fontfamily{msb}%
  \fontseries{m}\fontshape{n}\selectfont}
% Define command \tBbb as Blackboard bold
% to be used in text mode. Math mode is
% defined in file amssymb.sty.
```

```
%%% Cyrillic in text mode. %%%
```

```
\input{cyracc.def} % This file is needed
                  % for cyrillic fonts.

\newcommand{\tcyr}{\fontfamily{UWCyr}}
\selectfont{cyracc}
% Define font family only. Fontshape must
% be switched using \it or \sc commands.

\newcommand{\tcyrm}{\fontfamily{UWCyr}}
\fontseries{m}
\fontshape{n}\selectfont{cyracc}
% Define command \tcyrm as Univ. of
% Washington cyrillic to be used in
% text mode. This can also be achieved
% by typing \tcyr.

\newcommand{\tcyrb}{\fontfamily{UWCyr}}
\fontseries{b}
\fontshape{n}\selectfont{cyracc}
% Define command \tcyrb as bold Univ. of
% Washington cyrillic to be used in text
% mode. This can also be achieved by
% typing \bf\tcyr, provided \bfdefault is
% changed (see '{\AmS}-{\LaTeX} Version
% 1.1 User's Guide', Section 5.6,
% pp. 7--8 and Table 5, p. 14).
% {\AmS}Fonts give only wncyb ('b' for
% 'bold') and *not* wncbx ('bx' for
% 'bold extended').

\newcommand{\tcyrit}{\fontfamily{UWCyr}}
\fontseries{m}
\fontshape{it}\selectfont{cyracc}
% Define command \tcyrit as italic Univ.
% of Washington Cyrillic to be used in
% text mode. This can also be achieved
% by typing \it\tcyr.

\newcommand{\tcyrsc}{\fontfamily{UWCyr}}
\fontseries{m}
\fontshape{sc}\selectfont{cyracc}
% Define command \tcyrsc as small caps
% Univ. of Washington Cyrillic to be
% used in text mode. This can also be
% achieved by typing \sc\tcyr.
```

Acknowledgements

Thanks to Barbara Beeton, Bobby Bodenheimer, Guoying Chen, Michael Downes, George Greenwade, and Rainer Schöpf for their contributions to the document. They enhanced it substantially and did a superb job to make it error-free. The responsibility for any remaining bugs or inaccuracies falls entirely on me.

The information provided was compiled in the end of June 1992 and updated in February 1993.

◇ Rafał Żbikowski
 Control Group
 Dept. of Mechanical Engineering
 James Watt Building
 University of Glasgow
 Glasgow G12 8QQ
 Scotland, UK
 rafal@mech.gla.ac.uk

Letters

Response to A.G.W. Cameron

As organizer and editor of the proceedings of the colloquium "DeskTop Publishing in Astronomy and Space Sciences" organized at Strasbourg Astronomical Observatory in October 1991, I feel I must restate some facts and complete what has been written on the proceedings by A.G.W. Cameron in *TUGboat* 13, no. 4 (1992), pp. 489–490, even if the general conclusions are positive towards the volume and even if it is not my usual policy to react to reviews.

A review worthy of the name should have indicated the real purpose of meeting and give at least a brief description of its main organizational features (sessions).

It should have been made clear that this was not a *T_EX*-meeting and definitely not intended to discuss *T_EX*-nicalities. The main aim of the colloquium was to review the situation of desktop and electronic publishing in astronomy and space sciences, with the participation of publishers and scientific editors, and to initiate strategies for the future.

Since attendees were persons using potentially all kinds of systems, only broad guidelines were given for their camera-ready copy (CRC) contributions. I have been pleased that most contributors stuck as much as possible to the indications and I can only praise the publisher, World Scientific, for achieving an outstanding top quality volume compare to what is often found on astronomy library shelves and compare to what I had sometimes to put together in my editorial activities.

Not a word is said about a survey carried out beforehand on the use of packages in our community. The results of this survey are now quoted in the physical and other communities. *T_EX* is indeed leading, but it would be a mistake to ignore the significant amount of people using other systems. If, since the colloquium, there have been more journals pushing for the use of *T_EX*, meetings of editors and publishers are shaping procedures allowing input from other systems than *T_EX*, be it only by designing policies accommodating SGML, ODA, not to speak of the HyTime hypertext standard just approved by ISO.

No word is said either on the special sessions centered on publishers, editors, as well as on information retrieval, the ultimate aim of EP. A paper by an author from the same institution as the reviewer is criticized at length and one could wonder whether there are not personal accounts being settled there.

The reviewer seems impressed by the publishing speed. What should we think then of the proceedings of the "Astronomy from Large Databases-II" conference held in September 1992 (more than 500 pages) which were already in our hands in December 1992, with the classical CRC technique (as was the case for the DTP colloquium in question here).

This is where one starts asking oneself serious questions about EP since another conference on the Space Telescope, organized in May 1992 through the same channels, has not seen its proceedings volume completed yet, using EP techniques. But of course, EP is not the only factor to be considered in a publishing venture: the toughness of the editors is also important.

And, once more, DTP/EP techniques have not their own justifications per se, but through what they allow afterwards with the substance of the information they are carrying. For the ALD-II conference, an ftp account was already set up at the end of September (the week after the meeting) with the contributions. This was a premiere in astronomy accommodating a whole spectrum of input types, including *T_EX/L_AT_EX* of course, and in spite of the fact that CRC documents were the basic material for the proceedings (machine-readable files were also delivered in most cases).

Those who know my activities are aware that I have been pushing DTP/EP techniques on both sides of the Atlantic. One must however always keep in mind the respective advantages and limitations of each system.

André Heck
Observatoire Astronomique
11, rue de l'Université
F-67000 Strasbourg, France

Abstracts

Die TExnische Komödie

Contents of Recent Issues

1/1992, 4. Jahrgang (Mai 1992)

Luzia DIETSCHÉ, [Editorial]; p. 3

A short statement commenting on the current issue.

- *Hinter der Bühne : Vereinsinternes*
[Behind the scenes : Club matters]; pp. 4–28:

Joachim LAMMARSCH, Grußwort
[Welcome message]; pp. 4–5

A short statement commenting on the DANTE'92 conference.

[Luzia DIETSCHÉ and Joachim LAMMARSCH],
Protokoll der 6. Mitgliederversammlung von
DANTE, Deutschsprachige Anwendervereinigung
TEx e.V. [Protocol of the 6th assembly of
members of DANTE]; pp. 5–22

This is the official report on the members' meeting held during DANTE'92. It starts with short accounts on the various hardware platforms and `german.sty` (as presented by the appointed coordinators), followed by a report on the situation of DANTE (change of bylaws, organization, activities). A report on the meeting of the BoD of TUG (Birmingham, Feb. 14–16, 1992) is followed by some criticism of TUG, and the claim that European groups, in particular DANTE, are more widely accepted and offer better services than TUG. The report of the treasurer is accepted. Under the heading 'Planning for the future—A successor to TEx' a project under the working title of 'Pi' (estimated costs: DM 500 000) is presented and accepted. The final topic discussed is the negotiations with TUG, mainly concerning the transfer of money. The blame for lack of progress is attributed to TUG.

Friedhelm SOWA, Kassenbericht für den Zeitraum
1.1.1991 – 31.12.1991 [The treasurer's report for
1991]; pp. 23–25

Reinhard ZIERKE, Stammtischbericht aus
Hamburg [Report on the regular meeting in
Hamburg]; pp. 25–26

This short report on the local group in Hamburg is mainly concerned with software distribution.

Markus ERLMEIER, Software-Verteilung Amiga
[The distribution of Amiga software]; pp. 26–28

This article describes the software available for the Amiga, and how it can be obtained.

- *Bretter, die die Welt bedeuten*
[The stage is the world]; pp. 29–48:

Werner BURKHARDT, Klassenarbeiten mit LATEX
[Classroom examinations with LATEX]; pp. 29–32

This article presents two LATEX style files (input and output, no source code) for teachers preparing classroom examinations.

Wilhelm BARTH and Helmut STEINER, Deutsche
Silbentrennung für TEx 3.1 [German hyphenation
for TEx 3.1]; pp. 33–35

SiSiSi (*SIChere SInnentsprechende Silbentrennung*, i.e., 'safe and meaningful hyphenation') is an algorithm addressing the hyphenation problem posed by German compound words. It uses a dictionary of about 8000 words and is implemented on VAX/VMS via a change file to TEx. (The system can be obtained by ftp from `eichow.tuwien.ac.at`. The method used is described in a paper in: *Angewandte Informatik* 1985, pp. 152–159.)

Martin WALLMEIER, Deutsches und internationales
BIBTEXing [German and international BIBTEXing];
pp. 35–38

The standard style files for BIBTEX do not support multilingual bibliographic entries. The BIBTEX style files `gerplain` (etc.) together with a special LATEX style (`bibgerm.sty`) described here overcome this difficulty. They use an additional field `language` to allow language dependent formatting of each item (e.g., switching off capitalization).

Fred SUMBECK, Verwaltung von Literaturdaten
[Managing bibliographic data]; pp. 39–46

LITVER is a utility program that takes bibliographic data from a DBASE file and automatically inserts them into references in footnotes and bibliographies of a LATEX document.

Jens SCHMIDT, METAFONT-Spielerei
[METAFONT recreation]; pp. 46–48

A short piece of METAFONT code is used to modify cm-fonts (as inch high sans serif bold extended) to produce an artistic font suitable for posters and giving a 'nervous' impression.

- *Was Sie schon immer über TeX wissen wollten ...* [What you always wanted to know about TeX ...]; pp. 49–50:

Luzia DIETSCHÉ, Anfängerfragen gesucht
[Looking for beginners' questions]; p. 49
A new service offers help for newbies.

Rainer SCHÖPF, Drei TeX-Rätsel — die Auflösung
[Three TeX puzzles — solutions]; p. 50

The solutions to three questions on TeX from the previous issue (vol. 3, no. 4) are explained.

- *TeX-Beiprogramm* [TeX co-features]; p. 51:

Luzia DIETSCHÉ, Bezugsquellen für TeX
[Sources for TeX]; p. 51

A list of addresses where TeX can be obtained.

- *Leserbrief(e)* [Letter(s)]; pp. 52–55:

Horst SZILLAT, DC-Fonts; pp. 52–54

The author agrees with the criticism of the concept underlying the DC-fonts (expressed in an earlier letter by Peter Schmitt, vol. 3, no. 2) and makes some alternative suggestions. Furthermore, he objects to file naming conventions that are shaped according to the requirements of the least convenient operating system.

Claus LANGHANS, Serverzugang
[Access to servers]; p. 54

This letter is concerned with the difficulties of mailbox access by those without net access.

Helmut WÖHRLE, Geräteunabhängigkeit
[Device independence]; p. 55

The author of this letter praises the device independence of TeX files.

Guido F. PAULI, Wissenschaftliche Publikationen und TeX [Scientific publications and TeX]; p. 55

The author points out that many publishers (except publishers of texts in mathematics or physics texts) do not know TeX and its possibilities. He asks for samples with which one could show them that TeX can also be used for other fields (e.g., chemistry).

- *Spielplan* [Repertory]; pp. 56–62:

The international and national calendar, and announcements of conferences (EuroTeX'92 — First announcement, One day LATEX V3 conference, TUG'92).

- *Adressen* [Addresses]; pp. 63–67:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed

to this issue, and the addresses of the coordinators in charge of the various hardware platforms.

2/1992, 4. Jahrgang (August 1992)

Luzia DIETSCHÉ, [Editorial]; p. 3

A short opening statement asking (among others) for addresses of services offering phototypesetting of dvi-files.

- *Hinter der Bühne : Vereinsinternes*
[Behind the scenes : Club matters]; pp. 4–6:

Joachim LAMMARSCH, Grußwort
[Welcome message]; pp. 4–5

A short report on the author's impressions from the TeX Users Group conference in Portland, his meeting with D.E. Knuth, and the meeting of the board of directors. In particular, he expresses his strong disappointment over the statements on NTS (New Typesetting System) made by Malcolm Clark in his opening address.

Friedhelm SOWA, Stammtisch in Duisburg
[Regular meeting in Duisburg]; pp. 5–6

A report on the first local meeting in Duisburg.

- *TeX-Theatertage*
[TeX theatre festival]; pp. 7–12:

Arne WELLSSOW, Bericht über DANTE'92 in Hamburg [Report on the DANTE'92 conference in Hamburg]; pp. 7–12

This report on the proceedings of the conference contains short abstracts of the tutorial and the lectures.

- *Bretter, die die Welt bedeuten*
[The stage is the world]; pp. 13–45:

Erich NEUWIRTH, Die Weiterentwicklung von TeX : Protokoll vom 27. März 1992 im Rahmen von DANTE'92 in Hamburg [The future development of TeX]; pp. 13–15

This is the protocol of a meeting during DANTE'92 dedicated to the project of developing a successor to TeX. It consists mostly of questions. A mailing list dedicated to this project is announced (nts-1@vm.urz.uni-heidelberg.de).

Frank MITTELBACH, LATEX3; pp. 15–22

A report on the origins and the current state of the LATEX3 project, containing some details of the guiding principles (internal language for style design and the context model).

Claus LANGHANS, Elektronische Kommunikation für Jedermann [Electronic communication for everyone]; pp. 22–26

An overview of the network possibilities available for private use in Germany.

Horst SZILLAT, *AMS-LATEX*; pp. 26–31

This overview describes the advantages and disadvantages of American Mathematical Society-LATEX, and how to install the package.

Hans-Hermann BODE, Neue BIBTEX-Style-Files: Die adaptable family [New style files for BIBTEX: The adaptable family]; pp. 31–41

The new BIBTEX styles (*aplain*, etc.) presented are modifications of the standard styles. They provide the possibility to change the layout (fonts used for specific fields) and the words and phrases inserted by the style (like names for months, editions, and chapters).

Friedhelm SOWA, Text und Bilder [Text and pictures]; pp. 41–45

This article describes the history and the usage of *picinpar*, a set of macros which allow the specification of a window in a paragraph that can be filled by arbitrary material (text or graphics).

- *Was Sie schon immer über TEXwissen wollten ...* [What you always wanted to know about TEX ...]; p. 46:

Joachim LAMMARSCH, Umgebungsreich zu klein? [DOS-environment too small?]; p. 46

A hint describing how to enlarge the space available for DOS environment variables.

- *TEX-Beiprogramm* [TEX co-features]; p. 47:

Joachim LAMMARSCH, Das Ärgernis [The cause of anger]; p. 47

The author expresses his annoyance over two cases of software distribution, namely, the TEX system distributed by TUG (too little credit is given to emTEX), and a disk included in a special issue of CHIP (no credit to DANTE).

- *Rezensionen* [Reviews]; pp. 48–49:

Reinhard ZIERKE, Das Vieweg LATEX-Buch: Eine praxisorientierte Einführung [The Vieweg LATEX book: A practical introduction]; pp. 48–49

- *Leserbrief(e)* [Letter(s)]; p. 50:

Two letters, one asking for a printer driver for the NECP6+ under UNIX, the other informing that the PCTEX manual is now distributed by TUG.

- *Spielplan* [Repertory]; pp. 51–58:

The international and national calendar, and announcements of conferences (Clausthal-Zellerfeld and EuroTEX'92).

- *Adressen* [Addresses]; pp. 59–63:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed to this issue, and the addresses of the coordinators in charge of the various hardware platforms.

3/1992, 4. Jahrgang (November 1992)

Luzia DIETSCHE, [Editorial]; p. 3

In her opening statement, the managing editor of the journal comments on the divergent format in which contributed articles reach her. One article was even in Word format!

- *Hinter der Bühne : Vereinsinternes*

[Behind the scenes : Club matters]; pp. 4–17:

Luzia DIETSCHE and Joachim LAMMARSCH, Protokoll der 7. Mitgliederversammlung von DANTE, Deutschsprachige Anwendervereinigung TEX e.V. [Protocol of the 7th assembly of members of DANTE]; pp. 4–17

This is the official report on the members' meeting held in Clausthal. It starts with short accounts on the various hardware platforms, software services, and *german.sty* (presented by the appointed coordinators), followed by a report on the situation of DANTE (2102 members by August 1992, organization, services).

Joachim Lammarsch gives a detailed report on his problems with the publisher of the journal 'CHIP' (Vogel Verlag, already mentioned in the previous issue, p. 47) who did not give credit for the disk provided by DANTE. In his report on the software distribution by DANTE he admits problems that still cause rather delayed delivery. Nevertheless, when he cites a letter of a complaining member, he classifies the attitude expressed as an example of a consumer's thinking which (unfortunately) spreads more and more. The feedback, he deplores, is usually negative—there is almost never a word of thanks.

Then Joachim Lammarsch gives reports on his meeting with D.E. Knuth in Stanford (on NTS), and on the meeting of the BoD of TUG in Portland. The next TEX Users Group conference will be held in Birmingham. Though he suggests cancelling the European conference to avoid reduced numbers of participants, he also rejects the possibility

to announce it as a joint TUG *and* European conference.

Friedhelm Sowa delivers a short report on the Portland conference. He deplores that—in contrast to European conferences—the participants did not gather socially in the evenings (with only one exception, a bowling event), and that—due to the high costs—only a few ‘new faces’ appeared.

- *Bretter, die die Welt bedeuten*
[The stage is the world]; pp. 18–36:

Joachim BLESER and Edmund LANG, Bilder in L^AT_EX-Dokumenten [Pictures in L^AT_EX documents.]; pp. 18–24

The macro package `picins.sty` allows the inclusion of pictures in L^AT_EX documents. Four environments produce a frame for the picture (a normal frame, a frame with rounded corners, a dashed frame, and a frame with a shadow). The pictures can be placed at the beginning of a paragraph (left or right), or between paragraphs. The inclusion of pictures prepared by external means is briefly discussed.

Bernd RAICHLE, Umlaute im BIBTEX-Stylebefehl MACRO [Accents and the BIBTEX command MACRO]; pp. 25–27

In a BIBTEX macro definition the replacement text has to be put between double quotes, and the text may not contain a double quote. This poses a problem for German macro definitions, e.g., if one wants to define a macro for ‘M\"arz’ (the German name for March). The author discusses the disadvantages of various solutions, and proposes a new method.

Oliver SCHURR, Die documentstyle-Familie SCRIPT [The documentstyle family SCRIPT]; pp. 27–28

The author reports his positive experiences with the SCRIPT package for L^AT_EX which allows easy adaption to German layout conventions, and which provides a very flexible `letter` style.

Peter WILLADT, Anregungen hinsichtlich der Softwareverteilung von DANTE e.V. [Suggestions concerning the DANTE software distribution]; pp. 29–30

This article contains a checklist for authors of software packages: What should they observe when preparing a release?

Dieter JURZITZA, WANDEL.EXE—ein vielfältig nutzbarer Formatkonverter [WANDEL.EXE—a format converter with many uses]; pp. 31–32

WANDEL is a program that converts WordStar (3 and 5) and ASCII files to `TeX`, and that performs various conversions between WordStar 3 and 5, Tempus, and ASCII files.

Thomas ESKEN, Kurzer Nachtrag zur `TeX`-Benutzeroberfläche Tx1 [A short addendum to the `TeX` user interface Tx1]; pp. 33–34

This article describes the features of Tx1 (now version 4.10), a free `TeX` shell, and an accompanying utility DVIChk. The package is also available in English.

Christoph GÜLICHER, Autoaufkleber [Car stickers]; pp. 34–35

This note gives the L^AT_EX code (using `portland.sty`) to produce a sticker with the inscription ‘High `TeX` Car’.

W.F. KUCHEL and P.H. MUUS, Kalender [Datebook]; pp. 35–36

KALENDER is a shareware program that produces a .dvi file for printing out a datebook containing personal entries (addresses, birthdays, etc.).

○ *Was Sie schon immer über TeX wissen wollten ...* [What you always wanted to know about `TeX` ...]; pp. 37–38:

Luzia DIETSCHE, Blocksatz beim Abbildungs-/Tabellentitel [Paragraph shape in the caption of a figure or table]; pp. 37–38

It is shown how the `article` style has to be modified to provide captions with hanging indentation after the first line.

○ *TeX-Beiprogramm* [`TeX` co-features]; pp. 39–40:
Ekkehard HUNDT, Nachdrucke der Knuth-Bände [Reprints of the books by Knuth]; pp. 39–40

This is a list of the latest editions of the books from *Computers & Typesetting*. It also informs that the error logs of these books can be obtained free of charge.

○ *Rezensionen* [Reviews]; pp. 41–43:
Joachim LAMMARSCH, A Beginner’s Book of `TeX`; pp. 41–43

This is a review of the English translation (by Silvio Levy) of the book by Raymond Seroul. It mainly consists of a description of the contents.

- *Leserbrief(e)* [Letter(s)]; pp. 44–45:

Four letters, one pointing out an anglicism in an article by Friedhelm Sowa (who used *Paragraph* instead of *Absatz*) in the previous issue, one looking for a used phototypesetter, one wanting help with the use of Devanagari fonts, and one offering used *TEX* material.

- *Spielplan* [Repertory]; pp. 46–51:

The international and national calendar, and announcements of conferences (TUG'93 call for papers : World Wide Window on *TEX*).

- *Adressen* [Addresses]; pp. 52–55:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed to this issue, and the addresses of the technical advisors in charge of the various hardware platforms.

4/1992, 4. Jahrgang (Dezember 1992)

Luzia DIETSCHE, [Editorial]; p. 3

In her opening statement the managing editor of the journal comments on the difficulties posed by *PiCTEX* and by the *LATEX picture* environment when contributed articles have to be converted to the A5 format used by the journal (usually from A4 format).

- *Hinter der Bühne : Vereinsinternes*

[Behind the scenes : Club matters]; pp. 4–8:

Joachim LAMMARSCH, Grußwort

[Welcome message]; pp. 4–6

In his column the president of DANTE addresses various organizational questions (payment of membership dues, organizing an office, software and books distributed by DANTE). One remarkable fact mentioned is that about half of the *TEX* books delivered by Addison-Wesley are out of date—the record being a book on *METAFONT 1.0!*

Lothar MEYER-LERBS, *TEX* auf dem Mac
[*TEX* on the Mac]; p. 6

This note lists sources and current versions of Oz*TEX*, *Textures*, and Direct*TEX*.

Markus ERLMEIER, *TEXnische Software für den Amiga* [*TEXnical software for the Amiga*]; pp. 6–8

This article explains how DANTE members may order *TEX* software for the Amiga.

- *Bretter, die die Welt bedeuten*

[The stage is the world]; pp. 9–34:

Jürgen GLÖCKNER, Erzeugung virtueller Fonts
[Generating virtual fonts]; pp. 9–16

Everybody has heard of virtual fonts—but who uses them? This article explains in detail some of the basics of generating virtual fonts. For the following three examples complete step-by-step instructions are given: (i) replacing the digits of one font by those of another one; (ii) using PostScript-\special to obtain grey digits; (iii) adapting the font coding scheme of one font to that of another one (in this case: for the DVIPS drivers of ArborText and of Tomas Rokicki). Articles like this may help that virtual fonts—a powerful, but yet often neglected tool—are more commonly used.

Joachim DIPPEL, Neues zum *LATEX*-Seitenlayout:
Der Ourhead Style [News on *LATEX* page layout:
The ourhead style]; pp. 16–20

The author explains how the definitions of headlines and footlines in *article* style can be modified to realize a complicated page layout, e.g., framed pages, with (framed) multi-line page heads and page footers.

Stefan BREUER, Anmerkungen zum \footnote-Befehl [Remarks on the \footnote command]; pp. 20–22

The author describes an ad-hoc solution in which he modified the *LATEX* \footnote command to obtain a format demanded by a publisher: footnotes as a list (with numbers) and indented text.

Werner BURKHARDT, Schaubilder mathematischer Funktionen mit *PiCTEX* [Diagrams of mathematical functions using *PiCTEX*]; pp. 22–27

This *PiCTEX* tutorial shows (using the sine function as an example) how the graph of a function can be plotted in a labelled coordinate system. The final code is obtained by refinement in several steps. (It must be remarked, however, that using only five values of the function—at $-\pi$, $-\pi/2$, 0, $\pi/2$, and π —is too crude, even if the plotted curve might look quite all right.)

Joachim BLESER and Edmund LANG,
Balkendiagramme in *LATEX*-Dokumenten
[Column charts in *LATEX* documents]; pp. 28–31

This article describes the use of *bar.sty*, a *LATEX* style file that uses the *picture* environment to draw column charts. The columns can be drawn 2- or 3-dimensional. Eight different patterns can be used to shade the columns.

Dieter JURZITZA, Schaltbilder mit LATEX
[Circuit diagrams with LATEX]; pp. 32–34

A LATEX file `e_symbol` defines various elements which help to draw circuit diagrams in the `picture` environment.

- *Was Sie schon immer über TeX wissen wollten ...* [What you always wanted to know about TeX ...]; p. 35:

Luzia DIETSCHE, Fette mathematische Akzente [Bold mathematical accents]; p. 35

LATEX's `\boldmath` cannot be used to produce accented bold letters because it only switches to bold math fonts which do not contain the accents. One has to use `\bf` instead.

- *TeX-Beiprogramm* [TeX co-features]; p. 36:

Markus ERLMEIER, PasTeX — neue Version in Planung [PasTeX — a new version is in preparation]; p. 36

PasTeX is a public domain implementation of TeX for the Amiga. Its author Georg Heßmann plans a new version, and this is a call for suggestions and critical remarks.

- *Rezensionen* [Reviews]; pp. 37–40:

Ernst BRATZ, Beinahe gut — ein subjektiver Test von Scientific Word [Almost good — a subjective test of Scientific Word]; pp. 37–38

Scientific Word (from TCI Software) is a Windows-based package that includes an editor (which exports special LATEX code), a LATEX compiler, a previewer, and various printer drivers. The reviewer observes two main deficiencies: the LATEX compiler is extremely slow, and it is difficult (or, for German documents, sometimes even impossible) to import existing documents. Positively mentioned are formula editing facilities, and (German and English) spellchecking.

Lothar MEYER-LERBS, Ganz unTeXnische Bücher [Completely non-TExnical books]; pp. 38–40

The author provides pointers to some books of (not exclusively) typographical interest: a reprint of a collection of Chinese stories, edited 1944 by Jan Tschichold; the books published by Schumacher-Gebler, Munich, which are produced with great care, usually using traditional typesetting methods; and a visually designed book which was typeset using TeX (*Life Cast: Behind the Mask*).

- *Spielplan* [Repertory]; pp. 41–51:

The international and national calendar, and announcements of conferences (DANTE'93 in Chemnitz: March 9–12; 'TeX for non-American languages—METAFONT in Theory and Practice' near London, instead of in Glasgow: April 6–8; and TUG93 in Birmingham: July 26–30).

- *Adressen* [Addresses]; pp. 52–55:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed to this issue, and the addresses of the technical advisors in charge of the various hardware platforms.

(compiled by Peter Schmitt)

Calendar

1993

- May 11 **T_EX and Arabic Script**, INALCO (Institut National des Langues et Civilisations Orientales), Paris. Co-sponsored by GUTenberg and INALCO. For information, contact Yannis Haralambous (yannis@gat.citilille.fr).
- May 25 UK T_EX Users' Group, Chichester, England. Visit to John Wiley & Sons Ltd. Host: Geeti Granger. For information, contact David Murphy (D.V.Murphy@computer-science.birmingham.ac.uk).
- Jun 3 T_EX-Stammtisch at the Universität Bremen, Germany. For information, contact Martin Schröder (115d@calf.zfn.uni-bremen.de; telephone 0421/628813).
- Jun 6–10 Society for Technical Communication, 40th Annual Conference, Dallas, Texas. For information, contact the Society headquarters, 901 N. Stuart St., Suite 904, Arlington, VA 22203-1854. (703-522-4114; Fax: 703-522-2075), or the Conference Manager, Binion Amerson (aba@oc.com).
- Jun 9 **TUG Course:** T_EX for Publishers, New York City.
-
- TUG Courses, San Diego, California**
- Jun 7–11 Modifying L^AT_EX Style Files
- Jun 14–18 Beginning/Intermediate T_EX
- Jun 21–25 Advanced T_EX and Macro Writing
-
- Jun 10 11th NTG Meeting, "From Font to Book", Royal Dutch Meteorological Institute, De Bilt, Netherlands. For information, contact Theo Jurriens (taj@astro.rug.nl).
- Jun 15 T_EX-Stammtisch in Duisburg, Germany. For information, contact Friedhelm Sowa (tex@ze8.rz.uni-duesseldorf.de; telephone 0211/311 3913).
- Jun 16–18 SSP93—15th Annual Meeting: "What Will We Be Tomorrow?", Society for Scholarly Publishing, Hyatt Crystal City, Washington, D.C. For information, contact SSP, 10200 W 44th Ave., #304, Wheat Ridge, CO 80033 (303-422-3914, Fax: 303-422-8894).
- Jun 21 T_EX-Stammtisch in Bonn, Germany. For information, contact Herbert Framke (Herbert_Framke@BN.MAUS.DE; telephone 02241 400018).
- Jun 25 **TUGboat Volume 14, 2nd regular issue:** Mailing date (tentative).
- Jun 30 T_EX-Stammtisch, Hamburg, Germany. For information, contact Reinhard Zierke (zierke@informatik.uni-hamburg.de; telephone (040) 54715-295).
- Jul 1 T_EX-Stammtisch at the Universität Bremen, Germany. (For contact information, see Jun 3.)
- Jul 19 T_EX-Stammtisch in Bonn, Germany. (For contact information, see Jun 21.)
- Jul 20 T_EX-Stammtisch in Duisburg, Germany. (For contact information, see Jun 15.)
- Jul 28 T_EX-Stammtisch, Hamburg, Germany. (For contact information, see Jun 30.)
-
- TUG '93 Conference, Aston University, Birmingham, U.K.**
- For additional information concerning courses, see the announcement following this Calendar.*
- Jul 20–24 Beginning/intermediate T_EX
- Jul 20–24 Advanced T_EX and macros

Jul 21–23	METAFONT — logos	Sep 23–24	TUG Course: Book and Document Design with \TeX , Boston, Massachusetts
Jul 22–23	Book Design in \TeX	Sep 29	\TeX -Stammtisch, Hamburg, Germany. (For contact information, see Jun 30.)
Jul 22–23	Beyond Computer Modern — using other fonts in \TeX	Oct 7	\TeX -Stammtisch at the Universität Bremen, Germany. (For contact information, see Jun 3.)
Jul 22–23	\TeX and PostScript	Oct 18–22	TUG Course: Beginning/Intermediate \TeX Santa Barbara, California
Jul 24–25	Doing more with \LaTeX 2.09	Oct 18	\TeX -Stammtisch in Bonn, Germany. (For contact information, see Jun 21.)
Jul 26–30	TUG Annual Meeting: "World Wide Window on \TeX ". For information, contact tug93-enquiries@vax.rhbnc.ac.uk or the TUG office.	Oct 19	\TeX -Stammtisch in Duisburg, Germany. (For contact information, see Jun 15.)
Jul 31	Using the \TeX family for setting maths	Oct 27	\TeX -Stammtisch, Hamburg, Germany. (For contact information, see Jun 30.)
Jul 31 – Aug 1	\LaTeX 2.09 Style files		
Jul 31 – Aug 3	Intensive \LaTeX		
Aug 2–4	More \TeX — output routines		
Aug 2–5	Intensive \LaTeX		
Aug 2–6	METAFONT — fonts		

Aug 5	\TeX -Stammtisch at the Universität Bremen, Germany. (For contact information, see Jun 3.)
Aug 16	\TeX -Stammtisch in Bonn, Germany. (For contact information, see Jun 21.)
Aug 17	TUGboat Volume 14, 3rd regular issue: Deadline for receipt of technical manuscripts (tentative).
Aug 23–27	TUG Course: Intensive \LaTeX , Ottawa, Canada
Aug 25	\TeX -Stammtisch, Hamburg, Germany. (For contact information, see Jun 30.)
Sep 2	\TeX -Stammtisch at the Universität Bremen, Germany. (For contact information, see Jun 3.)
Sep 14	TUGboat Volume 14, 3rd regular issue: Deadline for receipt of news items, reports (tentative).
Sep 20	\TeX -Stammtisch in Bonn, Germany. (For contact information, see Jun 21.)
Sep 21	\TeX -Stammtisch in Duisburg, Germany. (For contact information, see Jun 15.)

TUG Courses, Santa Barbara, California

Oct 25–29	Intensive \LaTeX
Nov 1–5	Advanced \TeX and Macro Writing
Nov 8–9	Practical SGML and \TeX

Nov 4	\TeX -Stammtisch at the Universität Bremen, Germany. (For contact information, see Jun 3.)
Nov 10	TUG Course: SGML and \TeX for Publishers, New York City
Nov 12	TUG Course: \TeX for Publishers, Washington, DC
Nov 15	\TeX -Stammtisch in Bonn, Germany. (For contact information, see Jun 21.)
Nov 16	\TeX -Stammtisch in Duisburg, Germany. (For contact information, see Jun 15.)
Nov 18	12 th NTG Meeting, "(I)A \TeX user environment", Oce, Den Bosch, Netherlands. For information, contact Gerard van Nes (vannes@ecn.nl).
Nov 23	TUGboat Volume 14, 3rd regular issue: Mailing date (tentative).

Nov 24	T _E X-Stammtisch, Hamburg, Germany. (For contact information, see Jun 30.)
Dec 2	T _E X-Stammtisch at the Universität Bremen, Germany. (For contact information, see Jun 3.)
Dec 20	T _E X-Stammtisch in Bonn, Germany. (For contact information, see Jun 21.)
Dec 22	T _E X-Stammtisch, Hamburg, Germany. (For contact information, see Jun 30.)
Dec 21	T _E X-Stammtisch in Duisburg, Germany. (For contact information, see Jun 15.)

1994

- Apr 11–15 Four conferences, Darmstadt, Germany:
- EP94, Electronic Publishing, Document Manipulation and Typography (for information, contact ep94@gmd.de);
 - RIDT94, Raster Imaging and Digital Typography (for information, contact ridt94@irisa.fr);
 - TEP94, Teaching Electronic Publishing (for information, contact ltsdyson@reading.ac.uk);
 - PODP94, Principles of Document Processing (for information, contact podp94@cs.umd.edu). Deadline for submission of papers: 15 August 1993

For additional information on the events listed above, contact the TUG office (805-963-1338, fax: 805-963-8358, email: tug@math.ams.org) unless otherwise noted.

**Courses to be held in conjunction with
TUG '93**

It has become customary at TUG annual meetings for a variety of T_EX-related courses to be offered during the preceding and following weeks. This year's meeting, 26–30 July 1993, at Aston University, Birmingham, U.K., is no exception. The following courses have been scheduled.

1. Beginning/Intermediate T_EX, July 20–24; instructor: Michael Doob
2. Intensive L_AT_EX, July 31–August 3 and August 2–5 (two courses); instructor: Malcolm Clark
3. Advanced T_EX and macros, July 20–24; instructor: Chris Rowley
4. T_EX output routines, August 2–4; instructor: Philip Taylor
5. L_AT_EX 2.09 style files, July 31–August 1; instructor: Sue Brooks
6. METAFONT—logos, July 21–23; Erik-Jan Vens
7. METAFONT—fonts, August 2–6; instructor: Yannis Haralambous
8. Using the T_EX family for setting maths, including use of A_MS-L_AT_EX, July 31; instructor: Rosemary Bailey
9. Book design in T_EX, July 22–23; instructor: Philip Taylor
10. Doing more with L_AT_EX 2.09, July 24–25; instructors: Michel Goossens and Frank Mittelbach
11. Beyond Computer Modern—using other fonts in T_EX, July 22–23; instructor Yannis Haralambous
12. T_EX and PostScript, July 22–23; instructor: Sebastian Rahtz

All courses will be held at Aston University. For further details, contact

Carol Hewlett
Computer Service, LSE
London, WC2A 2AE, UK
e-mail: HEWLETT@LSE.AC.UK
phone: +44 71-955 7926
fax: +44 71-242 0392

TEXniques

Publications for the TEX Community

Available now:

1. **VAX Language-Sensitive Editor (LSEDIT)**
Quick Reference Guide for Use with the LATEX Environment and LATEX Style Templates by Kent McPherson
2. **Table Making – the INRSTEX Method** by Michael J. Ferguson
3. **User's Guide to the IdxTEX Program** by R. L. Aurbach
4. **User's Guide to the GlocTEX Program** by R. L. Aurbach
5. **Conference Proceedings, T_EX Users Group Eighth Annual Meeting, Seattle, August 24–26, 1987, Dean Guenther, Editor**
6. **The PICTEX Manual** by Michael J. Wichura
7. **Conference Proceedings, T_EX Users Group Ninth Annual Meeting, Montréal, August 22–24, 1988, Christina Thiele, Editor**
8. **A Users' Guide for T_EX** by Frances Huth
9. **An Introduction to LATEX** by Michael Urban
10. **LATEX Command Summary** by L. Botway and C. Biemesderfer
11. **First Grade T_EX** by Arthur Samuel
12. **A Gentle Introduction to T_EX** by Michael Doob
13. **METAFONTware** by Donald E. Knuth, Tomas G. Rokicki, and Arthur Samuel
14. **A Permuted Index for T_EX and LATEX** by Bill Cheswick

In production:

15. **EDMAC: A Plain T_EX Format for Critical Editions**
by John Lavagnino and Dominik Wujastyk

**TEX Users Group
P.O. Box 869
Santa Barbara, CA 02940, U.S.A.**

Late-Breaking News

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by e-mail, on diskette, and was also retrieved from remote sites by anonymous ftp. In addition to text and various code files processable directly by *T_EX*, the input to this issue includes METAFONT source code. More than 60 files were used directly to generate the final copy; over 100 more contain earlier versions of articles, auxiliary information, and records of correspondence with authors and referees. These numbers represent input files only; .dvi files, device-specific translations, and fonts (.tfm files and rasters) are excluded from the total.

Most articles as received were fully tagged for *TUGboat*, using either the plain-based or L^AT_EX conventions described in the Authors' Guide (see *TUGboat* 10, no. 3, pages 378–385). Copies of the macros were provided to several authors via e-mail; however, the macros have also been installed at labrea.stanford.edu and other good archives, and an author retrieving them from an archive will most likely get faster service. Of course, the TUG office will provide copies of the macros on diskette to authors who have no electronic access.

Just over 50% of the articles in this issue are in L^AT_EX, accounting for slightly under 45% of the pages. In organizing the issue, attention was given to grouping bunches of plain or L^AT_EX articles, to yield the smallest number of separate typesetter runs, and the least amount of handwork pasting together partial pages. This also affected the articles written or tagged by the staff, as the conventions of *tugboat.sty* or *ltugboat.sty* would be chosen depending on what conventions were used in the preceding and following articles; no article was changed from one to the other, however, regardless of convenience.

Font work was required for the article by Wong on Chinese pinyin (p. 8); font complications forced deferral of two articles until the next issue.

The graphics in the article by Comenetz (p. 25) were accomplished in P_CT_EX.

Test runs of articles were made separately and in groups to determine the arrangement and page numbers (to satisfy any possible cross references). A file containing all starting page numbers, needed

in any case for the table of contents, was compiled before the final run. Final processing was done in 5 runs of *T_EX* and 2 of L^AT_EX, using the page number file for reference.

The following articles were prepared using the plain-based *tugboat.sty*:

- the book review, Arvind Borde, *Mathematical T_EX by Example*, page 20.
- all articles in the Macros column except for "The bag of tricks".
- abstracts of the *T_EXnische Körnödie*, page 71.
- the TUG calendar, page 77.
- summary of courses to be held at TUG 93, page 79.
- these Production notes.
- "Coming next issue".

Output

The bulk of this issue was prepared at the American Mathematical Society from files installed on a VAX 6320 (VMS) and *T_EX*'ed on a server running under Unix on a Solbourne workstation. Output was typeset on the Math Society's Compugraphic 9600 Imagesetter, a PostScript-based machine, using the Blue Sky/Y&Y PostScript implementation of the CM fonts, with additional fonts downloaded for special purposes.

No pasteup of camera-ready items or illustrations was required for this issue.

The output devices used to prepare the advertisements were not usually identified; anyone interested in determining how a particular ad was prepared should inquire of the advertiser.

Coming Next Issue

Implementing the extended T_EX layout using PostScript fonts

Now that virtual fonts and 8-bit input are part of the general *T_EX* repertoire, the means are available to incorporate PostScript fonts in a more straightforward manner than was previously possible. Sebastian Rahtz demonstrates how the Cork layout can be implemented using PostScript fonts.

ET — A TeX-compatible editor for MSDOS computers

After observing that a handwritten equation in a colleague's draft of a scientific paper is much easier to read than the corresponding TeX source, at least if the equation is complicated and the handwriting is neat, John Collins has created an editor called ET ("Edit TeX") that satisfies these requirements:

- visual representation of the most common mathematical constructs, including greek letters, sub-/superscripts, and built-up fractions, showing them as mathematics, not as TeX control sequences;
- Keyboard, as opposed to menu, entry of symbols and other constructs;
- Almost "standard" (I^A)TeX code generated, and it is simple to enter ordinary (I^A)TeX commands that the editor does not handle;
- Ability to import/export existing (I^A)TeX files;
- Adequate performance even on low-grade IBM clones.

Essential NFSS2

Sebastian Rahtz offers a user's view of the New Font Selection Scheme, version 2. In this article he describes the reasons for using the NFSS; the differences a user will encounter between NFSS and old L^ATeX; what it will be like installing and using NFSS2; some special situations in mathematics; and an overview of the advanced NFSS2 commands for describing new fonts.

A Pragmatic Approach to Paragraphs

Appalled by the number of Underfull \hbox messages he has encountered in TeX documents received from the outside world, Philip Taylor has developed a technique for resetting the parameters that control the setting of paragraphs to achieve the best possible results under any particular conditions.

Index of Advertisers

88	American Mathematical Society
88	ArborText
Cover 3	Blue Sky Research
89	ETP (Electronic Technical Publishing)
90, 91	Kinch Computer Company
84	Micro Programs, Inc.
80	TeX Users Group
92	Y&Y

Institutional Members

The Aerospace Corporation,
El Segundo, California

Air Force Institute of Technology,
Wright-Patterson AFB, Ohio

American Mathematical Society,
Providence, Rhode Island

ArborText, Inc.,
Ann Arbor, Michigan

ASCII Corporation,
Tokyo, Japan

Beckman Instruments,
Diagnostic Systems Group,
Brea, California

Brookhaven National Laboratory,
Upton, New York

Brown University,
Providence, Rhode Island

California Institute of Technology,
Pasadena, California

Calvin College,
Grand Rapids, Michigan

Carleton University,
Ottawa, Ontario, Canada

Centre Inter-Régional de
Calcul Électronique, CNRS,
Orsay, France

CERN, *Geneva, Switzerland*

College Militaire Royal de Saint
Jean, *St. Jean, Quebec, Canada*

College of William & Mary,
Department of Computer Science,
Williamsburg, Virginia

Communications
Security Establishment,
Department of National Defence,
Ottawa, Ontario, Canada

Cornell University,
Mathematics Department,
Ithaca, New York

E.S. Ingenieros Industriales,
Sevilla, Spain

Elsevier Science Publishers B.V.,
Amsterdam, The Netherlands

European Southern Observatory,
*Garching bei München,
Federal Republic of Germany*

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

Florida State University,
Supercomputer Computations
Research, *Tallahassee, Florida*

GKSS, Forschungszentrum
Geesthacht GmbH,
*Geesthacht, Federal Republic of
Germany*

Grinnell College,
Computer Services,
Grinnell, Iowa

Grumman Aerospace,
Melbourne Systems Division,
Melbourne, Florida

GTE Laboratories,
Waltham, Massachusetts

Hungarian Academy of Sciences,
Computer and Automation
Institute, *Budapest, Hungary*

Institute for Advanced Study,
Princeton, New Jersey

Institute for Defense Analyses,
Communications Research
Division, *Princeton, New Jersey*

Intevep S. A., *Caracas, Venezuela*

Iowa State University,
Ames, Iowa

Los Alamos National Laboratory,
University of California,
Los Alamos, New Mexico

Louisiana State University,
Baton Rouge, Louisiana

MacroSoft, *Warsaw, Poland*

Marquette University,
Department of Mathematics,
Statistics and Computer Science,
Milwaukee, Wisconsin

Masaryk University,
Brno, Czechoslovakia

Mathematical Reviews,
American Mathematical Society,
Ann Arbor, Michigan

Max Planck Institut
für Mathematik,
Bonn, Federal Republic of Germany

National Research Council
Canada, Computation Centre,
Ottawa, Ontario, Canada

Naval Postgraduate School,
Monterey, California

New York University,
Academic Computing Facility,
New York, New York

Nippon Telegraph &
Telephone Corporation,
Software Laboratories,
Tokyo, Japan

Observatoire de Genève,
Université de Genève,
Sauverny, Switzerland

The Open University,
Academic Computing Services,
Milton Keynes, England

Personal T_EX, Incorporated,
Mill Valley, California

Politecnico di Torino,
Torino, Italy

Princeton University,
Princeton, New Jersey

Rogaland University,
Stavanger, Norway

Ruhr Universität Bochum,
Rechenzentrum,
*Bochum, Federal Republic of
Germany*

St. Albans School,
*Mount St. Alban, Washington,
D.C.*

Smithsonian Astrophysical
Observatory, Computation Facility,
Cambridge, Massachusetts

Space Telescope Science Institute,
Baltimore, Maryland

Springer-Verlag,
*Heidelberg, Federal Republic of
Germany*

Springer-Verlag New York, Inc.,
New York, New York

Stanford Linear
Accelerator Center (SLAC),
Stanford, California

Stanford University,
Computer Science Department,
Stanford, California

Texas A & M University, Department of Computer Science, <i>College Station, Texas</i>	University of Exeter, Computer Unit, <i>Exeter, Devon, England</i>	University of Stockholm, Department of Mathematics, <i>Stockholm, Sweden</i>
United States Military Academy, <i>West Point, New York</i>	University of Groningen, <i>Groningen, The Netherlands</i>	University of Texas at Austin, <i>Austin, Texas</i>
University of British Columbia, Computing Centre, <i>Vancouver, British Columbia,</i> <i>Canada</i>	University of Heidelberg, Computing Center, <i>Heidelberg, Germany</i>	University of Washington, Department of Computer Science, <i>Seattle, Washington</i>
University of British Columbia, Mathematics Department, <i>Vancouver, British Columbia,</i> <i>Canada</i>	University of Illinois at Chicago, Computer Center, <i>Chicago, Illinois</i>	Uppsala University, <i>Uppsala, Sweden</i>
University of California, Berkeley, Space Astrophysics Group, <i>Berkeley, California</i>	Universität Koblenz-Landau, <i>Koblenz, Federal Republic of</i> <i>Germany</i>	Villanova University, <i>Villanova, Pennsylvania</i>
University of California, Irvine, Information & Computer Science, <i>Irvine, California</i>	University of Maryland, Department of Computer Science, <i>College Park, Maryland</i>	Virginia Polytechnic Institute, Interdisciplinary Center for Applied Mathematics, <i>Blacksburg, Virginia</i>
University of California, Santa Barbara, <i>Santa Barbara, California</i>	Università degli Studi di Trento, <i>Trento, Italy</i>	Vrije Universiteit, <i>Amsterdam, The Netherlands</i>
University of Canterbury, <i>Christchurch, New Zealand</i>	University of Oslo, Institute of Informatics, <i>Blindern, Oslo, Norway</i>	Washington State University, <i>Pullman, Washington</i>
University College, <i>Cork, Ireland</i>	University of Salford, <i>Salford, England</i>	Wolters Kluwer, <i>Dordrecht, The Netherlands</i>
University of Crete, Institute of Computer Science, <i>Heraklio, Crete, Greece</i>	University of South Carolina, Department of Mathematics, <i>Columbia, South Carolina</i>	Yale University, Department of Computer Science, <i>New Haven, Connecticut</i>
University of Delaware, <i>Newark, Delaware</i>	University of Southern California, Information Sciences Institute, <i>Marina del Rey, California</i>	

FOR YOUR TEX TOOLBOX**CAPTURE**

Capture graphics generated by application programs. Make LaserJet images compatible with TEX. Create pk files from pcl or pcx files. \$135.00

texpic

Use texpic graphics package to integrate simple graphics—boxes, circles, ellipses, lines, arrows—into your TEX documents. \$79.00

Voyager

TEX macros to produce viewgraphs—including bar charts—quickly and easily. They provide format, indentation, font, and spacing control. \$25.00

FOR YOUR TEX BOOKSHELF**TEX BY EXAMPLE**

NEW!

Input and output are shown side-by-side. Quickly see how to obtain desired output. \$19.95

TEX BY TOPIC

NEW!

Learn to program complicated macros. \$29.25

TEX FOR THE IMPATIENT

Includes a complete description of TEX's control sequences. \$29.25

TEX FOR THE BEGINNER

NEW!

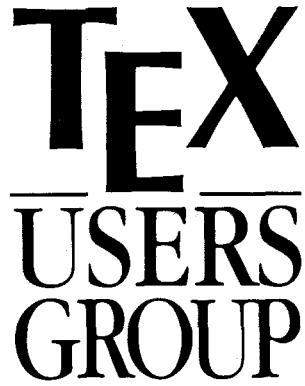
A carefully paced tutorial introduction. \$29.25

BEGINNER'S BOOK OF TEX

A friendly introduction for beginners and aspiring "wizards." \$29.95



Micro Programs Inc. 251 Jackson Ave. Syosset, NY 11791 (516) 921-1351



Complete and return this form with payment to:

TEX Users Group
Membership Department
P.O. Box 21041
Santa Barbara, CA 93121-1041
USA

Membership is effective from January 1 to December 31 and includes subscriptions to *TUGboat*, *The Communications of the TeX Users Group* and the TUG newsletter, *TeX and TUG News*. Members who join after January 1 will receive all issues published that calendar year.

For more information ...

Whether or not you join TUG now, feel free to return this form to request more information. Be sure to include your name and address in the spaces provided to the right.

Check all items you wish to receive below:

- Institutional membership information
- Course and meeting information
- Advertising rates
- More information on TeX

Correspondence unaccompanied by a payment should be directed to

TEX Users Group
P.O. Box 869
Santa Barbara, CA 93102 USA
Telephone: (805) 899-4673
Email: tug@Math.AMS.org

Individual Membership Application

Name _____

Institutional affiliation, if any _____

Position _____

Address (business or home (circle one)) _____

City _____

State or Country _____ Zip _____

Daytime telephone _____ FAX _____

Email addresses (*please specify networks, as well*) _____

I am also a member of the following other TeX organizations:

Specific applications or reasons for interest in TeX:

Hardware on which TeX is used:

Computer and operating system

Output device/printer

There are two types of TUG members: regular members, who pay annual dues of \$60; and full-time student members, whose annual dues are \$30. Students must include verification of student status with their applications.

Please indicate the type of membership for which you are applying:

- Regular @ \$60
- Full-time student @ \$30

Amount enclosed for 1993 membership: \$ _____

(Prepayment in US dollars drawn on a US bank is required)

- Check/money order payable to TeX Users Group enclosed
- Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____

TEX USERS GROUP

Complete and return this form with payment to:

TEX Users Group
Membership Department
P.O. Box 21041
Santa Barbara, CA 93121-1041
USA

Membership is effective from January 1 to December 31. Members who join after January 1 will receive all issues of *TUGboat* published that calendar year.

For more information ...

Correspondence

TEX Users Group
P.O. Box 869
Santa Barbara, CA 93102 USA
Telephone: (805) 899-4673
Email: tug@math.ams.org

Whether or not you join TUG now, feel free to return this form to request more information.

Check all items you wish to receive below:

- Course and meeting information
- Advertising rates
- More information on TEX

Institutional Membership Application

Institution or Organization _____

Principal contact _____

Address _____

City _____

State or Country _____ Zip _____

Daytime telephone _____ FAX _____

Email addresses (*please specify networks, as well*) _____

Each Institutional Membership entitles the institution to:

- designate a number of individuals to have full status as TUG individual members;
- take advantage of reduced rates for TUG meetings and courses for *all* staff members;
- be acknowledged in every issue of *TUGboat* published during the membership year.

Educational institutions receive a \$100 discount in the membership fee. The three basic categories of Institutional Membership each include a certain number of individual memberships. Additional individual memberships may be obtained at the rates indicated. Fees are as follows:

Category	Rate (educ./ non-educ.)	Add'l mem.
A (includes 7 memberships)	\$ 540 / \$ 640	\$50 ea.
B (includes 12 memberships)	\$ 815 / \$ 915	\$50 ea.
C (includes 30 memberships)	\$1710 / \$1810	\$40 ea.

Please indicate the type of membership for which you are applying:

Category _____ + _____ additional individual memberships

Amount enclosed for 1993 membership: \$ _____

Check/money order payable to TEX Users Group enclosed
(*payment is required in US dollars drawn on a US bank*)

Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____

Please attach a corresponding list of individuals whom you wish to designate as TUG individual members. Minimally, we require names and addresses so that TUG publications may be sent directly to these individuals, but we would also appreciate receiving the supplemental information regarding phone numbers, email addresses, TEX interests, and hardware configurations as requested on the TUG Individual Membership Application form. For this purpose, the latter application form may be photocopied and mailed with this form.

TEX CONSULTING & PRODUCTION SERVICES

North America

Abrahams, Paul

214 River Road, Deerfield, MA 01342;
(413) 774-5500

Development of TeX macros and macro packages. Short courses in TeX. Editing assistance for authors of technical articles, particularly those whose native language is not English. My background includes programming, computer science, mathematics, and authorship of *TeX for the Impatient*.

American Mathematical Society

P.O. Box 6248, Providence, RI 02940;
(401) 455-4060

Typesetting from DVI files on an Autologic APS Micro-5 or an Agfa Compugraphic 9600 (PostScript). Times Roman and Computer Modern fonts. Composition services for mathematical and technical books and journal production.

Anagnostopoulos, Paul C.

433 Rutland Street, Carlisle, MA 01741;
(508) 371-2316

Composition and typesetting of high-quality books and technical documents. Production using Computer Modern or any available PostScript fonts. Assistance with book design. I am a computer consultant with a Computer Science education.

ArborText, Inc.

1000 Victors Way, Suite 400, Ann Arbor,
MI 48108; (313) 996-3566

TeX installation and applications support. TeX-related software products.

Archetype Publishing, Inc.,

Lori McWilliam Pickert

P.O. Box 6567, Champaign, IL 61821;
(217) 359-8178

Experienced in producing and editing technical journals with TeX; complete book production from manuscript to camera-ready copy; TeX macro writing including complete macro packages; consulting.

The Bartlett Press, Inc.,

Frederick H. Bartlett

Harrison Towers, 6F, 575 Easton Avenue,
Somerset, NJ 08873; (201) 745-9412

Vast experience: 100+ macro packages, over 30,000 pages published with our macros; over a decade's experience in all facets of publishing, both TeX and non-TeX; all services from copyediting and design to final mechanicals.

Cowan, Dr. Ray F.

141 Del Medio Ave. #134, Mountain View, CA 94040; (415) 949-4911

Ten Years of TeX and Related Software Consulting, Books, Documentation, Journals, and Newsletters. TeX & LATEX macropackages, graphics; PostScript language applications; device drivers; fonts; systems.

Electronic Technical Publishing Services Co.

2906 Northeast Glisan Street, Portland, Oregon 97232-3295;
(503) 234-5522; FAX: (503) 234-5604

Total concept services include editorial, design, illustration, project management, composition and prepress. Our years of experience with TeX and other electronic tools have brought us the expertise to work effectively with publishers, editors, and authors. ETP supports the efforts of the TeX Users Group and the world-wide TeX community in the advancement of superior technical communications.

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

1101 San Antonio Road, Suite 413,
Mountain View, CA 94043-1002;
(415) 691-1126;
ogawa@applelink.apple.com

Specialist in fine typography, LATEX book production systems, database publishing, and SGML. Programming services in TeX, LATEX, PostScript, SGML, DTDs, and general applications. Instruction in TeX, LATEX, and SGML. Custom fonts.

Pronk&Associates Inc.

1129 Leslie Street, Don Mills, Ontario,
Canada M3C 2K5;
(416) 441-3760; Fax: (416) 441-9991

Complete design and production service. One, two and four-color books. Combine text, art and photography, then output directly to imposed film. Servicing the publishing community for ten years.

Quixote Digital Typography, Don Hosek

349 Springfield, #24, Claremont, CA 91711; (714) 621-1291

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.

TeXnology, Inc., Amy Hendrickson

57 Longwood Ave., Brookline, MA 02146;
(617) 738-8029

TeX macro writing (author of MacroTeX); custom macros to meet publisher's or designer's specifications; instruction.

Type 2000

16 Madrona Avenue, Mill Valley, CA 94941;
(415) 388-8873; FAX (415) 388-8865

\$2.50 per page for 2000 DPI TeX camera ready output! We have a three year history of providing high quality and fast turnaround to dozens of publishers, journals, authors and consultants who use TeX. Computer Modern, Bitstream and METAFONT fonts available. We accept DVI files only and output on RC paper. \$2.25 per page for 100+ pages, \$2.00 per page for 500+ pages.

Outside North America

TypeTeX Ltd.

Electronical Publishing, Battyány u. 14.
Budapest, Hungary H-1015;
(036) 11152 337

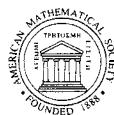
Editing and typesetting technical journals and books with TeX from manuscript to camera ready copy. Macro writing, font designing, TeX consulting and teaching.

Information about these services can be obtained from:

TeX Users Group

P. O. Box 869
Santa Barbara, CA 93102-0869
Phone: (805) 963-1388
Fax: (805) 963-8538

T_EX Publishing Services



From the Basic:

The American Mathematical Society offers you two basic, low cost T_EX publishing services.

- You provide a DVI file and we will produce typeset pages using an Autologic APS Micro-5 phototypesetter. \$5 per page for the first 100 pages; \$2.50 per page for additional pages.
- You provide a PostScript output file and we will provide typeset pages using an Agfa/Compugraphic 9600 imagesetter. \$7 per page for the first 100 pages; \$3.50 per page for additional pages.

There is a \$30 minimum charge for either service. Quick turnaround is also provided... a manuscript up to 500 pages can be back in your hands in one week or less.

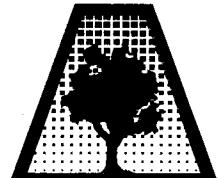
To the Complex:

As a full-service T_EX publisher, you can look to the American Mathematical Society as a single source for any or all your publishing needs.

Macro-Writing	T _E X Problem Solving	Non-CM Fonts	Keyboarding
Art and Pasteup	Camera Work	Printing and Binding	Distribution

For more information or to schedule a job, please contact Regina Girouard, American Mathematical Society, P. O. Box 6248, Providence, RI 02940, or call 401-455-4060.

NEW!



NEW!

ARBORTEXT INC.

- Silicon Graphics Iris or Indigo
- Solaris 2.1
- DVILASER/HP3
- Motif and OPEN LOOK Preview

**Complete T_EX packages
Ready to use, fully documented and supported.**

Also Available For: Sun-4 (SPARC), IBM RS/6000,
DEC/RISC-Ultrix, HP 9000, and IBM PC's

Call us for more information on our exciting new products!

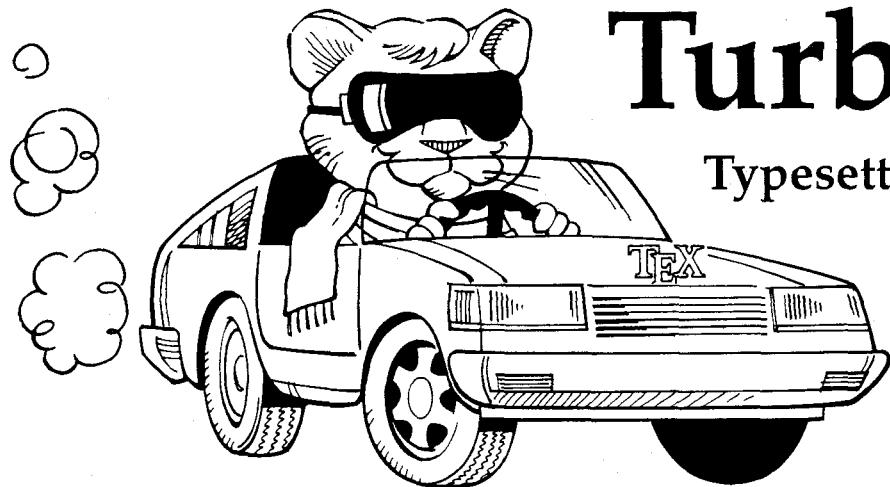
The solution is ETP.

$$\Delta \mathcal{P} = \sum_W \left[Q_{\text{IPR}} \int_{4-1-87}^{\infty} (D_p + D_m + D_s)^T + \epsilon(P_m - I_P) dt \right] \equiv \text{ETP}$$

ETP Services offers solutions to the problems facing the publishers of technical books and journals, with a complete array of composition-related services.

Electronic Technical Publishing Services Company

2906 N.E. Glisan Street
Portland, Oregon 97232
503-234-5522 •• FAX: 503-234-5604
mimi@etp.com



TurboTeX

Typesetting Software

Executables \$150
With Source \$300



THE MOST VERSATILE \TeX ever published is breaking new ground in the powerful and convenient graphical environment of Microsoft Windows: TurboTeX Release 3.1E. TurboTeX runs on all the most popular operating systems (Windows, MS-DOS, OS/2, and UNIX) and provides the latest \TeX 3.14 and METAFONT 2.7 standards and certifications: preloaded plain \TeX , \LaTeX , $\text{AMS-}\text{\TeX}$ and $\text{AMS-}\text{\LaTeX}$, previewers for PC's and X-servers, METAFONT, Computer Modern and \LaTeX fonts, and printer drivers for HP LaserJet and DeskJet, PostScript, and Epson LQ and FX dot-matrix printers.

■ Best-selling Value: TurboTeX sets the world standard for power and value among \TeX implementations: one price buys a complete, commercially-hardened typesetting system. *Computer* magazine recommended it as "the version of \TeX to have," *IEEE Software* called it "industrial strength," and thousands of satisfied users around the globe agree. TurboTeX gets you started quickly, installing itself automatically under MS-DOS or Microsoft Windows, and compiling itself automatically under UNIX. The 90-page User's Guide includes generous examples and a full index, and leads you step-by-step through installing and using \TeX and METAFONT.

■ Classic \TeX for Windows. Even if you have never used Windows on your PC, the speed and power of TurboTeX will convince you of the benefits. While the \TeX command-line options and $\text{\TeX}book$ interaction work the same, you also can control \TeX using friendly icons, menus, and

dialog boxes. Windows protected mode frees you from MS-DOS limitations like DOS extenders, overlay swapping, and scarce memory. You can run long \TeX formatting or printing jobs in the background while using other programs in the foreground.

■ MS-DOS Power, Too: TurboTeX still includes the plain MS-DOS programs. Virtual memory simulation provides the same sized \TeX that runs on multi-megabyte mainframes, with capacity for large documents, complicated formats, and demanding macro packages.

■ Source Code: The portable C source to TurboTeX consists of over 100,000 lines of generously commented \TeX , TurboTeX, METAFONT, previewer, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; Windows interface; and preloading, virtual memory, and graphics code, all meeting C portability standards like ANSI and K&R.

■ Availability & Requirements: TurboTeX executables for IBM PC's include the User's Guide and require 640K, hard disk, and MS-DOS 3.0 or later. Windows versions run on Microsoft Windows 3.0 or 3.1. Order source code (includes Programmer's Guide) for other machines. On the PC, source compiles with Microsoft C, Watcom C 8.0, or Borland C++ 2.0; other operating systems need a 32-bit C compiler supporting UNIX standard I/O. Specify 5-1/4" or 3-1/2" PC-format floppy disks.

■ Upgrade at Low Cost. If you have TurboTeX Release 3.0, upgrade to the latest version for just \$40 (ex-

ecutables) or \$80 (including source). Or, get either applicable upgrade free when you buy the AP- \TeX fonts (see facing page) for \$200!

■ No-risk trial offer: Examine the documentation and run the PC TurboTeX for 10 days. If you are not satisfied, return it for a 100% refund or credit. (Offer applies to PC executables only.)

■ Free Buyer's Guide: Ask for the free, 70-page Buyer's Guide for details on TurboTeX and dozens of \TeX -related products: previewers, \TeX -to-FAX and \TeX -to-Ventura/Pagemaker translators, optional fonts, graphics editors, public domain \TeX accessory software, books and reports.

Ordering TurboTeX

Ordering TurboTeX is easy and delivery is fast, by phone, FAX, or mail. Terms: Check with order (free media and ground shipping in US), VISA, Mastercard (free media, shipping extra); Net 30 to well-rated firms and public agencies (shipping and media extra). Discounts available for quantities or resale. International orders gladly expedited via Air or Express Mail.

The Kinch Computer Company

PUBLISHERS OF TURBOTeX
501 South Meadow Street
Ithaca, New York 14850 USA
Telephone (607) 273-0222
FAX (607) 273-0484

AP-TEX Fonts

TEX-compatible Bit-Mapped Fonts Identical to Adobe PostScript Typefaces

If you are hungry for new *TEX* fonts, here is a feast guaranteed to satisfy the biggest appetite! The AP-TEX fonts serve you a banquet of gourmet delights: 438 fonts covering 18 sizes of 35 styles, at a total price of \$200. The AP-TEX fonts consist of PK and TFM files which are exact *TEX*-compatible equivalents (including "hinted" pixels) to the popular PostScript name-brand fonts shown at the right. Since they are directly compatible with any standard *TEX* implementation (including kerning and ligatures), you don't have to be a *TEX* expert to install or use them.

When ordering, specify resolution of 300 dpi (for laser printers), 180 dpi (for 24-pin dot matrix printers), or 118 dpi (for previewers). Each set is on ten 360 KB 5-1/4" PC floppy disks. The \$200 price applies to the first set you order; order additional sets at other resolutions for \$60 each. A 30-page user's guide fully explains how to install and use the fonts. Sizes included are 5, 6, 7, 8, 9, 10, 11, 12, 14.4, 17.3, 20.7, and 24.9 points; headline styles (equivalent to Times Roman, Helvetica, and Palatino, all in bold) also include sizes 29.9, 35.8, 43.0, 51.6, 61.9, and 74.3 points.

The Kinch Computer Company
PUBLISHERS OF TURBOTEX
501 South Meadow Street
Ithaca, New York 14850
Telephone (607) 273-0222
FAX (607) 273-0484

Helvetica, Palatino, Times, and New Century Schoolbook are trademarks of Allied Linotype Co. ITC Avant Garde, ITC Bookman, ITC Zapf Chancery, and ITC Zapf Dingbats are registered trademarks of International Typeface Corporation. PostScript is a registered trademark of Adobe Systems Incorporated. The owners of these trademarks and Adobe Systems, Inc. are not the authors, publishers, or licensors of the AP-TEX fonts. Kinch Computer Company is the sole author of the AP-TEX fonts, and has operated independently of the trademark owners and Adobe Systems, Inc. in publishing this software. Any reference in the AP-TEX font software or in this advertisement to these trademarks is solely for software compatibility or product comparison. LaserJet and DeskJet are trademarks of Hewlett-Packard Corporation. *TEX* is a trademark of the American Math Society. TurboTEX and AP-TEX are trademarks of Kinch Computer Company. Prices and specifications subject to change without notice. Revised October 9, 1990.

Avant Garde	<small>Bold</small>
Avant Garde	<small>Bold Oblique</small>
Avant Garde	<small>Demibold</small>
Avant Garde	<small>Demibold Oblique</small>
Bookman	<small>Light</small>
Bookman	<small>Light Italic</small>
Bookman	<small>Demibold</small>
Bookman	<small>Demibold Italic</small>
Courier	
Courier	<small>Oblique</small>
Courier	<small>Bold</small>
Courier	<small>Bold Oblique</small>
Helvetica	
Helvetica	<small>Oblique</small>
Helvetica	<small>Bold</small>
Helvetica	<small>Bold Oblique</small>
Helvetica Narrow	
Helvetica Narrow	<small>Oblique</small>
Helvetica Narrow	<small>Bold</small>
Helvetica Narrow	<small>Bold Oblique</small>
Schoolbook	<small>New Century Roman</small>
Schoolbook	<small>New Century Italic</small>
Schoolbook	<small>New Century Bold</small>
Schoolbook	<small>New Century Bold Italic</small>
Palatino	<small>Roman</small>
Palatino	<small>Italic</small>
Palatino	<small>Bold</small>
Palatino	<small>Bold Italic</small>
Times	<small>Roman</small>
Times	<small>Italic</small>
Times	<small>Bold</small>
Times	<small>Bold Italic</small>
Zapf Chancery	<small>Medium Italic</small>
Symbol	$\Delta\Phi\Gamma\vartheta\Lambda\pi\Theta$
Zapf Dingbats	

Lucida Bright + Lucida New Math

The Lucida Bright + Lucida New Math

font set is the first alternative to Computer Modern complete with math fonts in ATM compatible Adobe Type 1 format. Lucida Bright + Lucida New Math will give your papers and books a bright new look!

The Lucida New Math

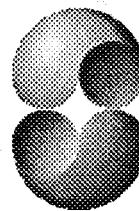
fonts (Lucida New Math Italic, Lucida New Math Symbol, Lucida New Math Extension and Lucida New Math Arrows) include the mathematical signs and symbols most used in mathematical and technical composition, including italic Greek capitals and lower case. The Lucida New Math fonts contain the math characters that are standard in the TeX math composition software, which, in its various forms, is one of the most popular mathematical composition packages used worldwide. In addition to the standard Computer Modern math font character sets, Lucida New Math fonts also include the characters in the American Mathematical Society (*AMS*) symbol fonts.

Switching to Lucida Bright + Lucida New Math is as easy as adding an input statement to the head of your TeX source file. Aside from four styles of each of the expected serifed, sans serif, and fixed width text faces, the font set also contains Lucida Blackletter, Lucida Calligraphy and Lucida Handwriting.

The Lucida Bright + Lucida New Math

font set is available in fully hinted ATM compatible Adobe Type 1 format for Macintosh, IBM PC compatibles, as well as Unix/NeXT.

We also carry the other font sets commonly used with TeX in fully hinted ATM compatible Adobe Type 1 format, but we are most excited about the new Lucida Bright + Lucida New Math fonts. The finest tools for working with scalable outline fonts in TeX are DVIPSONE and DVIWindo (on IBM PC compatibles).



Y&Y, Inc.
106 Indian Hill
Carlisle, MA 01741
(800) 742-4059 (508) 371-3286 (voice) (508) 371-2004 (fax)

"Lightning Textures works wonders. This has saved me many hours of corrections...I have three books to be completed and this new gadget certainly gives me courage." anonymous graduate student, University of Ottawa

"I find that Lightning sets type on my Quadra 700 faster than the VAX in the Lab can do it. It's a very nice program, indeed." John C. Allred, Los Alamos National Laboratory

"I used T_EX on a PC for years, but now that I've used Textures I wouldn't (want to) go back to that for anything." George Killough, Hendecagon Corporation

"A few weeks ago, I received a copy of Lightning Textures from you. I was rather skeptical when I ordered it. I could hardly believe that Lightning would work the way you described...But, after testing the program, I am fully satisfied, even more, I am excited about Lightning. Congratulations!" Dr. Bernd Fischer, Universität Hamburg

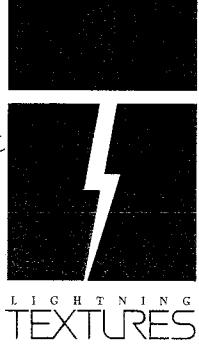
"I have Textures installed and running -- it's great! Who could prefer other implementations of T_EX, I wonder?." Mark Seymour, Springer-Verlag Heidelberg

"It's heaven." Ken Dreyhaupt, Springer-Verlag

"I know that Blue Sky Research did not intend Lightning Textures as a hacker's tool, but it sure can be used that way. It is strange how a simple speedup in the turnaround cycle (of about a factor of 10 to 100) changes your perspective. Lightning Textures is a joy to use." Victor Eijkhout, University of Tennessee

"I'll spread the word---Lightning Textures is great!" Prof. Anthony Siegman, Stanford University

"...a fantastic product and I can't imagine going back to an ordinary T_EX environment." Chuck Bouldin, Naval Research Laboratory



"I just typeset 300 pages of two-column heavy math in 7.45 minutes on a Mac IIci -- almost a 4X speedup (actually 3.8). I am very impressed. Good job." Stanley Rabinowitz, Editor of Index to Mathematical Problems

"I am absolutely amazed by Lightning Textures. It beats everything else hands down. I can't imagine what you could be putting in next." Kaveh Bazarghan, Focal Image

TUGBOAT

Volume 14, Number 1 / April 1993

	3 Addresses
General Delivery	5 Opening words / Christina Thiele
	7 Editorial comments / Barbara Beeton
Philology	8 Typesetting Chinese <i>pinyin</i> using virtual fonts / Wai Wong
Hardware / Systems	12 A multimedia document system based on T _E X and DVI documents / R. A. Vesilo and A. Dunn
Book Reviews	20 Arvind Borde, <i>Mathematical T_EX by Example</i> / Phil Taylor
	17 Michael Vulis, <i>Modern T_EX and Its Applications</i> / Jon Radel
Macros	23 The bag of tricks / Victor Eijkhout
	25 Anchored figures at either margin / Daniel Comenetz
	35 The \CASE and \FIND macros / Jonathan Fine
	40 Doing astronomical computations with T _E X: Making agendas / Jordi Saludes
	54 FIFO and LIFO sing the BLUes / Kees van der Laan
L_AT_EX	60 An update on the babel system / Johannes Braams
	62 Hacker's Guide to A _M SFonts and NFSS in the Context of L _A T _E X / Rafał Żbikowski
Letters	70 Response to A.G.W. Cameron / André Heck
Abstracts	71 Die T _E Xnische Komödie 1992, Heft 1-4
News & Announcements	77 Calendar
	79 Courses to be held in conjunction with TUG 93 (Aston University, Birmingham, U.K., 26-30 July 1993)
Late-Breaking News	81 Production notes / Barbara Beeton
	81 Coming next issue
TUG Business	83 Institutional members
Forms	85 TUG membership application
Advertisements	82 Index of advertisers
	87 T _E X consulting and production services