* * * * * * * * * * *

Site Reports

* * * * * * * * * * *

NEWS FROM ALL OVER

David Fuchs

Here's lots of news in no particular order:

TEX82. The code's in pretty good shape. We've been running with version 0.95 for six weeks, during which time we found three small bugs. At the same time, installing TEX found as many bugs in the VAX/VMS Pascal 2.0 compiler, and one bug in the DECSystem 10/20 microcode! This must mean something.

As I am writing this, Professor Knuth is testing version 0.96, which corrects the bugs, and adds a dozen or so new features, and about as many small improvements. Barring any further problems with microcode, by the time you read this, version 0.96 should be available through normal distribution channels. We expect to need only one more round of changes by the time the manual is complete.

Chapter 21 of the new manual ('Boxes') is currently being written. When Chapter 23 is done, we'll do a pre-print of the whole manual, and declare the current version of the code to be 1.0. Our tape distribution people [Ron and Maria Code, Data Processing Services, 1371 Sydney Drive, Sunnyvale CA 94087 (408-735-8006)] are willing to take orders for version 1.0, and they will hold them until everything is ready to go. Of course, you can order version 0.96 if you like, since it will be almost the same.

A new feature of the tape distribution is that you can order all the fonts at any of a number of different resolutions. Currently, we have 200, 220, 240, 260, 280 and 300 dot/inch versions available, but more will be forthcoming if the demand is great enough.

Right now, the distribution tape includes change files for TOPS-20 only. I hope to be able to add change files for other systems in the near future. On the other hand, it is fine for users of the same kind of systems to band together and handle it themselves. For instance, the VAX/UNIX port is up and running at a number of sites, and you should contact Rick Furuta for details on its distribution.

There have been several independent IBM ports, by Susan Plass at Stanford CIT, Roger Chaffee at SLAC, and Bruce Nemnich at MIT. The latter two have both discovered a method of saving a preloaded TEX under VM/CMS! John Johnson reports he has TEX82 running on his HP-1000, and is happy to talk to interested parties (he's getting output on his Epson printer). Bart Childs and Norm Naugle at Texas A&M are in the final stages of getting TEX82 running on their Data General MV8000. I'm sorry if I've left anyone out, my only excuse is that things are very hectic.

Conspicuously absent from the list is any 68000-based system. Our SUNs are just coming into operation, but it looks like the modifications done to the VAX/UNIX compiler to enable it to deal with TEX can be transferred directly to the SUN version of UNIX 4.1C BSD, and SUN Microsystems has promised to do so. Another encouraging sign: Apple is giving us a Lisa to try to put TEX on; it remains to be seen whether their compiler is good enough.

I've been working on a VMS port. Tangle, Weave, PLtoTF, TFtoPL and DVItype went into VMS 3.2 running Pascal 2.1 with little problem, since the new compiler has much improved I/O support. TEX82 on VMS currently is able to correctly run through the TRIP test file. The biggest installation problems were the half-dozen or so compiler bugs I ran into on the way and had to kludge around. These have been reported to Digital, so when the next release of the compiler rolls around everything should be beautiful.

Since the last TUGboat, we've added a new means of communication for the TEX community: an inter-network mailing list called TeXhax. Recipients of messages to TeXhax are on the ARPAnet, CSnet or UUCPnet. The mailing list has 70 members scattered around the country, and even in England and Germany. A number of the addresses are computer bulletin boards, so the readership is actually larger than 70. Most messages to date have dealt with TEX installation and portability questions, but I'm hoping that when the manual is completed and both TEX82 and CSnet are in wider use, TeXhax will become a forum for information about macros, output devices, fonts, etc. For those of you with no access to any network, we'll be summarizing newsworthy items from TeXhax in TUGboat (see my other article in this issue).

If you want to be added to TeXhax, send mail to `TEXHAX-REQUEST@SU-SCORE` on the ARPAnet, or `DRF@Stanford` on the CSnet, or `Shasta!DRF` on UUCP. Please include your network address, your real name and location, and CPU and output devices you're interested in. To send a message to everyone on TeXhax, send mail to `TEXHAX@Score` (or `..Shasta!Score!TEXHAX`, or to me, and I'll relay it).

One of the most exciting things going on is Leslie Lamport's work on his LaTeX system. This macro package makes TEX82 look a bit like Brian Reid's SCRIBE system, including such features as:

- A Scribe-like cross-reference system, where sections, theorems, etc. are given symbolic names by a `\label{foo}` command, and then referenced as "Section `\ref{foo}`".
- Scribe-like environment commands — e.g., for an enumerated list, one writes `\begin{enumerate}` ... `\end{enumerate}`. These will nest properly, and different levels will be numbered differently.
- A PICTURE environment, with picture-drawing commands.
- The ability to define different document styles.
- Automatic table of contents and list of figures.
- Flexible figure and formula numbering.
- Partial document compilation.

The system is now in "Beta Test" at a few installations, and the first preprint of the manual is done. Leslie has offered to speak about his system at this summer's TUG get-together, which is reason enough for everyone to attend! We hope to be able to put the entire thing on the distribution tape, and distribute the manual, before the TUG meeting.

We heard from Dr. Wolfgang Appelt of the Gesellschaft für Mathematik und Datenverarbeitung, a research organization in Germany. He reports that a meeting for TEX users in Germany was attended by people from more than a dozen installations, many of which were already running TEX82. We hope that this group will turn into the German chapter of TUG. Interested parties should contact Dr. Appelt at GMD, Postfach 12 40, Schloß Birlinghoven, D-5205 St. Augustin 1, Bonn. They are producing TEX82 output on a Perq/Canon printer, and are looking into the possibility of interfacing to the Hell-Digiset typesetter. Also, they hope to make use of Frank Liang's hyphenation-pattern-generator (which should be available on the distribution tape soon) to enable TEX82 to do German hyphenation.

I've heard from quite a few people with Imagen and Symbolics printers. It looks like prices are coming down a little, and I hope the new entry in the marketplace, the Lasergrafix machine based on the Xerox 2700 engine, will help continue the trend. I don't know any more about Lasergrafix, except that the folks at Texas A&M report that they are promising support for TEX output. Meanwhile, our HP 2680 has been connected to our Ethernet, and we hope to start seeing real TEX output from it soon.

On the phototypesetter front, things are heating up. The TEX project is looking for a new machine more appropriate for our needs than our Alphatype CRS. Autologic is promising that we'll be able to download our fonts to a Micro-5 if we buy one and write the code ourselves, but recently Compugraphic has become re-interested in TEX (perhaps taking the cue from HP and Apple). If they decide they're willing to let the 8600 be made Metafont-compatible, we'll be facing a hard decision. This has to get straightened out pretty quickly — we want to print the TEX manual on the new machine. Stay tuned.

A number of corporations around the country have inquired about the availability of TEX experts to hire, on a full-time or consulting basis. So far, I haven't been able to help them very much, but if you fit the bill, you might want to get in contact with me for more information.

Some people have encountered trouble with square roots on TEX82. Recall that TEX82 makes a different assumption about the position of the square root symbol within its box than TEX80 did. To get acceptable output with TEX82, you'll need up-to-date symbol and math extension fonts. (The reason for this change was to improve the alignment of the surd with the top rule, even in the face of rounding on different raster resolutions.)

Another item of note is that we're trying to improve the efficiency of TEX82. I should point out that it's none too shoddy as it is — the compiler being used in its development is very stupid when it comes to code generation, but we find that TEX82 is about the same speed as TEX80 in Sail, and it's quite a bit better than the old Pascal version. A good optimizing compiler, coupled with a reasonable disk I/O system should produce a fairly zippy TEX. In order to squeeze out lurking inefficiencies, however, we're conducting an experiment: A number of compilers support some form of statement execution profiling, where the user is told, after his program runs, how many times each statement was executed. With the aid of a little system wizardry, we're compiling such counts for TEX82, with the results cumulative over all users of TEX on the SU-AI computer. See if you can guess which Pascal statement was executed over 80 million times during the 2900 or so times TEX was run over the last 6 weeks (answer below). We're not just playing games, of course; we're trying to find where we should spend our time trying to speed up the code, by un-rolling loops, using macros instead of procedure calls, etc. Finally, there will be index entries in the published TEX code indicating the modules that are part of the inner loop, so that they can be hand-optimized on various systems if anyone cares to do so. The answer to the question above is CSPTR:=0 at the beginning of the GETNEXT procedure. By the way, of the almost 20000 lines of code in TEX82, 752 of them were not executed in over a month. Many of these statements are fatal error messages.

TEX on the HP-1000

Irene J Bunner and John D Johnson
JDJ Wordware, P.O. Box 354, Cupertino, CA 95015; (415) 985-3245

TEX82 for Hewlett-Packard HP-1000 Computer Systems has been implemented by JDJ Wordware. The code was obtained from Donald E. Knuth and the WEB programming system was used to perform the modifications required to make TEX82 run on this mini-computer. It is interfaced to an inexpensive ($500.00) Epson MX-80 dot matrix impact printer. A combination of automatic translation and hand editing have been used to create a set of fonts usable by this device.

HP-1000 mini-computers are 16 bit architecture machines with a real time, multiuser operating system. Programming is supported by a screen mode editor, a Pascal compiler, and a symbolic debugger. The standard addressing accommodates programs up to 64K Bytes, much too small for TEX. Extended addressing, supported by a virtual memory system, permits larger data sizes. Program segmentation permits larger code sizes. Heavy utilization of these two features enable us to fit TEX82 on the HP-1000.

The Epson MX-80 dot matrix impact printer is used in its high resolution graphics mode. In this mode it has a horizontal resolution of 120 dots per inch and a vertical resolution of 144 dots per inch. The dot size is larger than this resolution resulting in adjacent dots overlapping. A program was written that reads in the DVI file produced by TEX and special pixel files. It constructs the raster image for a complete page in memory before outputting to the printer. Interweaving rows of dots is required during this outputting. Eight horizontal rows of dots are printed, the paper is advanced by $\frac{1}{2}$ dot width then the eight interweaved horizontal rows of dots are printed. About six minutes are required to print a full $8\frac{1}{2}$ by 11 inch page.

Special raster descriptions for the fonts are required so the definition of pixel files was changed for this implementation. One of the changes is to a vertical raster instead of a horizontal raster. Even though the print head moves horizontally, the MX-80 is basically a vertical raster device. The eight dots it prints in parallel are arranged in a vertical row. Raster descriptions are changed from a 32 bit organization into a 16 bit organization to accommodate the word width of the HP-1000. The biggest change is to low resolution. Metafont was unavailable so the first approximation to the required pixel files was created by writing a program to automatically translate standard pixel files into the new format. This progam reads in a 240 dot per inch PXL file and produces a 120 dot per inch vertical raster pixel file. The small difference in the printer's horizontal and vertical resolution is ignored. Horizontal dimensions are accurate while vertical dimensions are compressed.

Translation to low resolution pixel files is achieved by mapping four pixels from the input file to a single pixel in the output file. If two or more of the input pixels are set, then the corresponding output pixel is set. Characters appear too black in these automatically created pixel files. The large dot size compared to the resolution causes the strokes to be too wide. Better looking characters are obtained by hand editing the fonts. A program was written which translates a pixel file into an ASCII file that contains a picture of each character built up using at signs and spaces. The system editor is then used to modify these pictures. Finally, another program translates from the ASCII file back into a pixel file. Hand editing a font requires several hours and several iterations.

Porting TEX to a small machine turned out to be quite a challenge. The first step was to bring up the Tangle program so that the WEB sources could be translated to Pascal. This was not too difficult. Porting TEX itself was much harder. Its data is too large, its code is too big and it has too many files for the standard Pascal runtime I/O system. The large data problem is handled by utilizing extended addressing which requires moving large arrays to the heap. WEB macros in the change file are used to do this. For each large array, the change file creates a pointer to the array as well as an entry in the initialization code which does a "new" on this pointer. Other macros change all the old accesses to the array into pointer dereferenced accesses.

The large code problem is solved by segmenting. This was very difficult because TEX is too large to be handled by the automatic multilevel segmenter. Special segmenting is used which allows the passing of value parameters between segments. Luckily, TEX uses only value parameters so no changes to parameter lists are required to accommodate the segmentation. A program was written to aid in segmenting TEX. First it reads and parses the Pascal source and builds a procedure call graph. Then this program is used interactively to place procedures in various segments. The goal is to minimize cross segment calls subject to an overall segment size restriction. Once the user is satisfied with the segmentation, this program builds the required source files to pass to the compiler as well as the segmentation files to pass to the linking programs. Several iterations were required before acceptable segmentation was achieved.

TEX declares more files than the standard Pascal runtime I/O system can handle. The main problem with this I/O package is its inability to use file buffers in the extended addressing area. To overcome this problem, a special runtime I/O system was written which can put file buffers in the heap. WEB macros are used to map standard I/O statements into calls to these special I/O routines.

TEX82 is now available on the HP-1000 family of computers. Currently, typesetting performance is roughly 1500 words per minute. This page was formatted using an A700 Series HP-1000 and output on an Epson MX-80 printer.

Unix TEX Site Report

Richard Furuta and Pierre MacKay[†]
Department of Computer Science
University of Washington

We'd like to take this opportunity to introduce ourselves to the *TUGboat* readership and report on new developments. We took over the Unix TEX site coordinator duties from Robert Morris in November, 1982. Since then, much of our TEX time has been devoted to TEX82, both on Unix and on Tops20. We have submitted a separate article in which we describe TEX at our particular site in more detail, but here, let us turn our attention to the state of TEX on Unix machines.

We are pleased to report that TEX82 is currently running on VAX/Unix, 4.1BSD, thanks to the efforts of Pavel Curtis of Cornell University and Howard Trickey of Stanford University. New versions of TEX82 have been coming up on Unix within a couple of weeks of their announcement at Stanford, so we are pretty much up to date. Howard and Pavel have written an article summarizing their experiences in porting TEX82 to Unix which follows this report. Details on how to obtain TEX82 for your own VAX are below.

Unfortunately, we have not heard of any versions of TEX82 for other flavors of Unix. Some interest has been expressed in porting TEX82 to the Sun workstation (running 4.2BSD) and some preliminary work is in progress. We have also heard from a person who wants to port TEX82 to a PDP-11 running 2BSD Unix, but as far as we know, he has not yet started working actively on it. If you're doing a port to one of these other machines or versions of Unix, please keep us posted.

We are trying to handle as much of our correspondence as possible through electronic mail. We can be reached from the Arpanet, from CSNet, and via uucp. Our Arpanet and CSNet addresses are Furuta@Washington and MacKay@Washington. On uucp, it's

...decvax!microsoft!uw-beaver!uw-june!furuta

and ...uw-june!mackay. An alternate uucp route is ...ucbvax!lbl-csam!uw-beaver....

[†]Our work is funded, in part, by a grant from Northern Telecom.

We also are maintaining an electronic mailing list, Unix-TEX@Washington, for discussions between Unix TEX sites. If you want to be included on this list, send Arpanet, CSNet, or uucp mail to Furuta.

TEX82 for Berkeley Unix, 4.1BSD

TEX82 is now available for sites running Berkeley Unix, version 4.1. We are now making available a "beta test" distribution which includes sources for TEX and WEB, libraries, fonts, and whatever device drivers we can obtain. At the time of writing, we have drivers for the Versatec, thanks to Carl Binding of our Department, and for the Imagen laser printer, thanks, again, to Pavel Curtis. We also hope to have a Symbolics laser printer driver in place before the distribution begins. If you have DVI 2 drivers for other devices, please send them to us and we'll happily include them in future distributions. Our fonts are now in the PXL format, 128 characters per font (the older arrangement), and with magnifications of 1.0, 1.2, and 1.3 included for most entries.

The present distribution will be the latest version of TEX82 available to us (presently 0.95). Note that versions before 1.0 are considered to be pre-releases so the language they define is subject to change. If you want to wait for 1.0, please let us know when you request your tape.

As the implementation uses a modified version of Berkeley's *pc* Pascal compiler, we will only be able to provide a complete distribution to those sites with source licenses for 4.1BSD. We will, however, try to accomodate those with binary licenses for 4.1BSD. Please make sure that you indicate clearly whether you have a source or a binary license for 4.1BSD.

The complete distribution presently occupies something on the order of 11 to 12 megabytes of disk storage on our machine. Of this, about 3 megabytes is used to store the fonts, 1 megabyte for the modified *pc* sources, and the remainder for TEX82, WEB, device drivers, and other parts of the system. Of course, not all of this needs to be retained on-line. The frugal site may be able to get by using only perhaps a half of this amount of disk space.

Tapes will be in tar format, blocked 20, and written at 1600 bpi, unless otherwise specified (we can also write 800 bpi).

To order, send a check for $50 (U.S. Funds) made to the University of Washington, documentation of the type of Unix license you hold, and your address to:

Richard Furuta
Computer Science, FR-35
University of Washington
Seattle, WA 98195

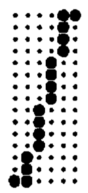
We would appreciate it if foreign sites could increase the amount of their check as appropriate to pay the added postal costs necessary for mailing the tape. We'd prefer not to receive purchase orders. Please contact us if that causes you problems and we'll try to set up alternate arrangements. If you have a CSNet, Arpanet, or uucp mail address, please include it—we'll add you to our mailing list.

The lower limit of low-res fonts

As noted above, we run TEX on a DECSystem 2060 as well as on the VAX. Since we do have use of the 2060, we occasionally run off PXL fonts for other Unix sites with METAFONT. (METAFONT is written in the SAIL programming language and so does not run on Unix machines. Indeed, use of METAFONT is pretty much limited to DECSystem-10's and DECSystem-20's.) We have recently received several requests for very coarse-grained PXL fonts at resolutions of 100 dots to the inch or less, which prompts a few observations.

100 dots to the inch is just acceptable for upright fonts, but rather chancy for slanted or italic fonts, and anything significantly lower is hopeless. There is not much point in asking for a PXL font at 0.36 magnification (72 dots to the inch) unless you simply want it as a rough guide for a pixel-editing program. At least half the characters are unidentifiable at that resolution. In the long run, the best results at any resolution under about 300 dots to the inch will require a stage of pixel editing. METAFONT is astonishingly good at making the critical decisions about difficult regions of a font character, but it is not perfect. At 100 dots to the inch, it is often overstrained. This point can best be appreciated by looking at an example from CMI10, produced at .5 magnification. The slant brings out all the worst problems of aliasing, and in this case produces a comically misleading result.

```
Char '135   Right bracket
Pixel Width 6 Pixel Height 15
X-offset 1 Y-offset 10
Raster Pointer 844
Width 0.27777 (2.7777 pt.)
```



(modified output from David Fuchs's PXLTYP)

The upper horizontal of the right square bracket is caught at a point where METAFONT's *rounding* operation shifts the entire lot of pixels one position to the right and as a result the bar ends up on the wrong side of the character. A pixel editor can fix this better than METAFONT can. There is no point in complicating the basic METAFONT character descriptions with statements that will handle such an extreme case. Even after editing, the italic and slant fonts (CMSnn, CMInn and CMTInn) will still look pretty crude. But most of the characters in roman, MATHEX and even symbol fonts are not too bad. Anyway, until the price and availability of 200 pixel/inch bit-mapped displays comes to be very different from what it is now, we must live with what we can get.

* * * *

Porting TEX to VAX UNIX

Pavel Curtis and Howard Trickey

Over the past few months, the authors have independently ported TEX82 to run under Berkeley's VAX/Unix[†] (version 4.1BSD). One of us (Pavel) subsequently merged our ideas to produce the VAX/Unix distribution of TEX announced in this issue's Unix TEX Site Report. Both implementations required small changes to Berkeley's *pc* compiler. This note describes the problems encountered in making these changes and in bringing up TEX itself.

TEX porters are aware that the capabilities of the available Pascal compiler will almost totally determine how difficult the port will be. Unhappily, the *pc* compiler has more deficiencies than one might wish. This is somewhat understandable, since it was designed for instructional use and not for the preparation of production software. On the plus side, the documentation was good enough to allow us to anticipate most of the problems.

[†]Unix and VAX are registered trademarks of Bell Laboratories and Digital Equipment Corporation, respectively.

Also, we seem to be the exception to "Fuchs' Law" that TeX uncovers at least one compiler bug for every compiler tried—our problems were caused by deficiencies, not bugs. This meant that there were few unexpected problems, and we were able to do the initial port in surprisingly little time (one or two weeks).

The most significant shortcoming of *pc* was also the one major place where Professor Knuth departed from Jensen and Wirth's Pascal and from the proposed standard: a default clause in the **case** statement. It has been suggested that the best way to handle this problem is to write a program to translate the **case** statements used in TeX to **case** statements without default clauses. This is actually fairly difficult to do properly, since it entails discovering the type of the case expression (character, integer, or integer subrange) and then forming a succinct test for the gaps between the expressions that actually occur as case labels. Since the *pc* compiler already generates code to check for an unmentioned value, the best possible translation would result in doing this work twice. The old PTeX implementation made a half-hearted attempt, by using an editor script to collect most of the information needed, but one still had to do the translation manually. We considered it imperative that bringing up a new version of TeX be fast and easy, enabling all installations to track bug fixes and added features.

The last thing one normally resorts to in order to circumvent a language deficiency is that horror of horrors: compiler modification. No one except the author or maintainer of a compiler really understands the implications of a given change to the code, so it is easy to introduce obscure bugs. Also, the compiler modifier has to be wary of new versions of the compiler from its maintainer. Having said this, we must now say that we both came to the conclusion, in this case, that compiler modification was the answer. Examination of the then current code showed that *pc* was already trapping the default case in order to have an error message printed. It was immediately obvious how to give control to another statement instead, and it was hard to see how any bugs could be introduced. Since so few changes were needed, the new-compiler-version problem was not too daunting, so the deed was done.

Compiler modification was definitely *not* the answer for the next problem, memory packing. To avoid wasting a large amount of memory, we needed to define the types *quarterword* and *halfword* to contain 8 and 16 bits, respectively. It is not possible to convince *pc* to pack 0..65535 into 16 bits or 0..255 into 8 bits. While it might not be too difficult to get the memory allocator to do this, the code generator could easily be a different question. Fortunately, the compiler *will* pack −128 .. 127 and −32768 .. 32767 into the proper sizes and TeX has been written so that these ranges can be easily used in the all important *memory_word* record. A similar problem occurs with the non-character *byte_files*, but here one must be careful that the bits actually written to the files look like the range 0 .. 255 has been used (otherwise, device independent output files and TeX font metric files can't be transported between sites). So, for example, instead of writing a value of 200 to a file, we write $200 - 256 = -56$.

In spite of the above, we would still waste a tremendous amount of memory if we could not find a way to fit the type called *glue_ratio* into 32 bits. Usually a *glue_ratio* is made the same as **real**, but *pc* puts **reals** into 64 bits, and has no provision for any kind of "short" **reals**. Fortunately, the VAX hardware has a 32-bit floating point data type, so the following trick could be used: Pascal is told that a *glue_ratio* is a (32-bit) integer, but we really store floating point numbers there; external procedures (written in C) are then used to convert back and forth between those pseudo-**reals** and *pc*'s 64-bit variety.

The rest of the changes necessary were more conventional. There were a few places where TeX uses variables names that *pc* uses for special things (e.g., *input*, *text*, *time*, and *date*), but the WEB system provided an easy way to fix them. The *pc* runtime system doesn't provide for recovery from file opening errors, or for closing a file immediately. It was easy to modify a few of the run-time library procedures to make files behave the way TeX assumes they will. Specifically, the following changes were made:

- Opening an input or output file that isn't there causes the "end-of-file" flag to be set true or false, respectively, rather than being a fatal error.
- Reading past end-of-file is silently ignored, rather than being a fatal error.
- The first **get** from the terminal is now ignored.
- Provision is made to explicitly close a file. The *pc* run-time system as supplied simply closes files on scope exit.

The alternative to changing the run-time library would be to do all input/output with external C procedures, but this would have meant changing every place in the code that used **get, put, read, write**, tests for end-of-file, etc. Macros would have helped to some extent, but it is hard to see how to convert a file variable argument into information that conveys the type of the file. Also, one has to be on guard for a varying number of arguments. Adding onto this the need to simulate Pascal's conventions about text files, it just seemed easier to make the run-time library changes.

One particularly irritating thing about *pc* is that it doesn't accept keywords which are written using uppercase characters, even though that is required by the standard! (More precisely, there is a flag allowing for upper case keywords to be used but this switch also turns off all of the other extensions Berkeley *did* make to the Jensen and Wirth standard.) There were a number of places in TANGLE that had to be changed in order to prevent the conversion to uppercase. A subtle bug can show up if one is not careful in making this change: when TANGLE does constant folding, it looks for keywords like MOD and DIV in the output buffer. The easiest place to convert to lowercase is before these comparisons, and if the comparisons are not changed an incorrect program may result.

Another problem we had with TANGLE was getting it to put the statement

```
#include "ext.h"
```

in the Pascal output. This statement is needed to tell *pc* about externally compiled procedures. The double quote marks in the statement could not be emitted by TANGLE, and there was no way to force the '#' to be just after a line break. We introduced primitives into WEB that eventually evolved into the "verbatim Pascal string" and the "force Pascal line break" primitives now in the standard WEB language.

The amount of time it takes to compile TEX is a major annoyance. Using an otherwise unloaded VAX 11/780, it takes approximately an hour of real time to create a runnable TEX from the WEB source code. Back in the real world, one of us has experienced upwards of a five hour wait when other people were using the machine. Now that the port is complete, this is no longer such a bother. On those infrequent occasions when a new version of TEX is being installed, one can simply have the compilation done in the middle of the night.

For regular use, it appears necessary to have a version of the program which has the PLAIN macros preloaded. Unfortunately, UNIX is not as amenable to this idea as some other systems, notably TOPS-20. We were able, however, to retrieve a program from the net.sources group on USENET which will do the trick. The program, called "undump", takes a core-image and the executable file which produced it and constructs a new executable file incorporating all of the data areas of the core-image. After some involved machinations and close perusal of the order of file opens or closes within TEX, a procedure was derived for preloading any macro set. This saves an average of 30 seconds or so of start-up time over the non-preloaded version.

Some statistics regarding the port are in order. The production version compiles into about 235,000 bytes of code. This doesn't include the large data arrays, which need not be allocated until runtime. At least 250,000 bytes of data area are needed in addition to the code to get a usable TEX. In a memory-rich system it is not unreasonable to allocate 500,000 bytes for this purpose. It is possible, as mentioned above, to save a version of TEX that has its data areas preloaded with fonts and macros, but one wouldn't want to have too many such versions; they are on the order of 750,000 bytes long each. As for running speed, a sample six-page paper (single-spaced, 10-point type, using a page of macros and a couple of fonts on top of PLAIN) took 44 cpu seconds to format using the preloaded-with-PLAIN version of the program. For comparison, the same paper took 37 cpu seconds on the SAIL DEC-10 computer used to develop TEX.

With regard to output drivers for UNIX, there are not many yet available. One of us (Howard) has a driver written in C for the Xerox Dover printer "press" format, adapted from a program that handled the old (version 1) DVI files. The new program handles either version. Pavel has written one for the Imagen/Canon ImPrint Laser Printer, also in C.

Looking back on the port, it must be said that the combination of the WEB system and Professor Knuth's careful coding has resulted in quite a portable program. It now takes less than a day to bring up a new version of TEX and check that it passes the nefarious TRIP test. It is rarely necessary to touch the changes we've made, since the new features and bug fixes usually don't occur in the "system-dependent" parts. As a result, TEX is a program that can easily stay up to date and incorporate the latest bug fixes at many sites on many hosts. That, combined with the care taken to ensure that the output will be the exactly the same from host to host, means that we have a solid basis for portable documents.

TEX at the University of Washington: Tops-20, Unix, Versatec, and the Monolithic

Pierre MacKay and Richard Furuta[†]
Department of Computer Science
University of Washington

We would like to take this opportunity to describe the TEX environment in the Department of Computer Science at the University of Washington.

We presently run TEX82 on a DECSystem-2060 and on a VAX11/780, located in our Computer Science Laboratory. Our primary output device has been an elderly Versatec printer/plotter which prints on 11" wide paper. The Versatec is shared by TEX, **troff**, and Scribe users. We expect to receive a laser printer shortly.

Since we have the use of the DEC-2060 and of the Arpanet, we have had the opportunity to copy working versions of TEX82 directly from the SCORE machine at Stanford and to prepare our documentation using TEX82 even before the problems involved in creating the Unix version of TEX were entirely solved. We therefore gave a high priority to putting together drivers for our Versatec. This effort was complicated by the fact that while the version 2 DVI files were created on the DEC-20, the Versatec was connected to the 780's Unibus, and we had no controller to drive the Versatec directly from the DEC-20 in any case.

We decided to begin by modifying a magnetic tape file transfer system which we had developed for TEX80 output. Although slow, this system had the distinct advantage that it supplied bit-mapped rasters to the Versatec at an even rate and avoided the extreme variations in toner density which can appear if the paper races past the print head to skip over white space. Our first, very provisional output driver, DVItoVRT, did all the necessary translation for the Versatec, and wrote full-width Versatec rasters, ten to a block, on magnetic tape on the DEC-20. We then read the tape on the VAX11-780 and printed the rasters on the Versatec. Generally, we could fit twenty or thirty pages worth of information onto a 2400' tape reel. DVItoVRT still exists, but it has never been entirely finished. It boasts the version number 0.8, but that may be something of an exaggeration. In any case it works. It is dreadfully slow, but it works, though only for vertically oriented output on roll-paper.

The DVItoVRT driver is based on the DVItype processor which was made available concurrently with TEX82, and it preserves all possible elements of DVItype. The addition to DVItype which may be of most general interest is the part that reads the PXL fonts. DVItype gets all the font information it needs from the TFM (TEX Font Metric) files associated with each font, but a driver program must take account of the structure of the font itself. Our programs read PXL files as described by David Fuchs in *TUGboat*, Volume 2, No. 3, pages 8-12; they allow for scaling either in the TEX input itself, or at the time when the DVI file is interpreted for the output device. The most practical use of scaling is the latter. As a general habit, we format all TEX material at true size and expand it to a 1.3 magnification in the driver program. A reduction of 77%, which is available on a several photocopying machines, restores true dimensions and very much improves the apparent sharpness of each typeface.

Since DVItoVRT does the necessary bit-pushing to create a Versatec raster in as nearly standard Pascal as we can manage, it is infuriatingly slow. Processing and writing a page of WEB output to tape takes about 30 seconds of elapsed time on a moderately loaded TOPS-20 system, which is enough to discourage all but the most devoted user. Fortunately, in December, BNR Inc. lent us a Monolithic Systems Corp. processor, which is intended to be used as an intelligent controller for the Versatec. (For a description of earlier uses of the Monolithic and of the support programs for earlier versions of TEX, see Phil Sherrod and Alan Wright. "TEX Support Programs." *TUGboat*, Volume 2, No. 1, pages 17-19.) Except on the rare occasions when its font memory fills up, and a new font has to be written over one of the old fonts, the Monolithic allows us to drive the Versatec just about as fast as it can physically be driven. Sherrod and Wright reported some improvements that they made at Vanderbilt University, but we have not felt the need to consider those yet. At present we are content with a two-step process based on software developed at Stanford some years ago involving (1) translation from DVI format to an intermediate work file (VER) format, and (2) shipment of the VER command and data file to the Monolithic processor. We did modify the format of the VER file to allow us to use the same second pass with both DVI 1 and DVI 2 and we updated the software to use PXL files. These intermediate work files (VER files) are queued and deleted automatically after use. This can be an inconvenience, since they must be totally regenerated again if needed, but they are so large that we cannot afford to leave them on the disk.

[†]Our work is funded, in part, by a grant from Northern Telecom.

We wrote a new program, **DVI2VER**, to handle the first pass for DVI 2 files. This program, based (like **DVItoVRT**) on **DVItype**, is now fully debugged for vertical output on continuous roll paper. We have not yet decided what to do about rotated output on fan-fold paper. The one thing we do not want to do is go back to bit-pushing in standard Pascal. The regular use of 1.3 magnification and **1300PXL** fonts makes fan-fold paper rather impractical in any case. We have left the necessary hooks in the program so that rotated output could be added, but there has been no decision when or whether we will hang anything on those hooks.

At present, the intermediate work file makes no attempt to conform with BigEndian conventions. It is very much an artifact of a 36-bit environment. We have worked out a coding system which would fit very nicely into a 32-bit environment, but at present we do not know of any users who could take advantage of it. The 32-bit coding would actually result in a slightly larger work file if the TEXed material included a large number of tall characters (e. g., most of the characters in **MATHEX** fonts), but it would result in a slightly smaller work file on a page with short rule segments. In any case, we will soon need to modify our present **VER** file format again to allow for the new 256 character **PXL** file format and to take advantage of the larger number of fonts which may be defined in TEX82.

Once we had our output from version 2 **DVI** working nicely, we were left with one more anomaly: our METAFONT still produced proof-mode output using version 1 **DVI**. We rewrote our copy of **MFOUT.SAI** for the new conventions, and now our entire TEX environment is consistent with TEX82. (When you try to produce postamble values to simulate TEX output, you have to remember the discrepancy between the TEX82 *scaled point* which is $1/(2^{16})$th of a printer's point, and the **PXL** font FIX, which is $1/(2^{20})$th of the design size. Otherwise you get some very strange dimensions.) We have also modified our METAFONT to use DEC's GIGI terminal as the "drawdisplay" output device rather than Stanford's Datadisc terminal. At present, our principal concern with METAFONT is the production of a set of *Naskhi* characters for an Arabic Script enhancement to TEX82.

* * * * * * * * * * * *

VAX/VMS

Monte C. Nichols
Sandia National Laboratory
Livermore, California

Last issue I reported that we expected to see TEX82 running on VAX/VMS systems by late 1982. Boy was I wrong. One problem has been the wait for the new VMS version of Pascal, although that is no longer a problem for most of us. Additionally, the group at Oregon Software has been unable to find spare time to devote to bringing up TEX82 (note that their work on TEX has been on an uncompensated basis — Yes, you can get something for nothing, Virginia, it just takes longer). Several others have been working on the implementation but have been thwarted by other problems. David Fuchs (Stanford) has just reported success using VMS 3.2 and Pascal 2.1; see page 14. Hopefully the new version of Pascal will soon reside on many VMS machines (it was delivered in Italy long before it was available here on the West coast) and we will all be in business.

There should be several other articles in this issue written by VMS folks. In addition, I have received several letters very recently that contain leads for further articles of interest to the VMS community. Hopefully they will find their way into the next issue of TUGboat.

David Fuchs has very kindly made available many more fonts in .tfm and .pxl format. They are for 200dpi devices and the .pxl files exist in magnifications of 1000, 1100, 1200, 1300, 1400, and 1500. These files are presently being unpacked from tape and will be sent soon to Oregon Software where they can be distributed to you. These fonts will work for the most recent version of TEX now being distributed by OS as well as the version of TEX82 that we all long to see. In addition, I still have the files sent me courtesy of AMS which supplement those from D. F. but which exist only as METAFONT files and thus need to be "treated" by the METAFONT program on a DEC 10 or 20 machine.

I want to again encourage anyone who develops a spooler for devices other than those presently available on VMS to send them to me or to Oregon Software so they can be made available to the rest of the VMS community.

* * * * * * * * * * * *

TEX AT CALMA R&D

David Krapp

At Calma R&D in San Diego, we produce approximately five thousand pages of technical documentation every six months. The majority of this work is in the form of user manuals for our computer-aided design systems. Until now, the typesetting of these books was done by a commercial typesetter, at considerable expense to the company. The typeset copy is then mailed to our corporate publications office for printing.

Late last year, after a brief familiarization and testing period, a decision was made to use TEX to produce camera-ready copy of all of our documentation. Lower costs, faster typesetting time, and keeping control of the entire production process in-house were major factors in this decision. The conversion to TEX was to occur within our normal delivery schedule, necessitating very fast and concentrated work to produce this large volume of material.

Work began immediately on the creation of a suitable macro package that allows us to create many of the complex layouts used in our books with relative ease. Lynne Price, TEX wizard from our Sunnyvale office, was (and is) instrumental in creating this package, as well as advising me on debugging, creation of new macros and generally soothing frazzled nerves. We now have an excellent table of contents generator, with an index generator just around the corner. Most of our problems resulted from trying to duplicate the current typeset layout of the books, instead of simply redesigning them in a way that's easy for TEX. Happily, we have not encountered anything that absolutely *can't* be done (somehow).

Our macro package was designed so that our typists could easily insert control sequences in the text of the manuals. I then review the manuals to insert any complicated layouts, such as table alignments and the syntax descriptions of computer commands, for example:

$$FOO\left\{\begin{array}{l}\text{P P P}\\ \text{P var}\\ \text{VPA P ws}\\ \text{MAX Xs}\\ \text{MAX Ys}\\ \text{MAX Zs}\\ \text{PIR IS}_{arc,con}\\ \left[\text{MAG}\left\{\begin{array}{l}PP\\ \text{ALL}\\ \text{UPP}\\ \text{DWN}\end{array}\right\}[\text{IS}_{plnsrf}]^{*}\right]\end{array}\qquad \text{[[vec] [ADD] OTR lst]} + \right\}C/R$$

(One book had nearly 400 such syntaxes.) Finally, I debug and run the input file to generate the manual.

In February of this year (two months before our production deadline) we switched from the original TEX to the newer version (using TFM files). We now run TEX on both our VAX-11/780 and 730 under VMS, with output available on a Versatec V-80 (used for draft copy) and a Symbolics LGP-1 Laser Graphics printer (for camera copy). As this is written, we are pushing hard to meet the deadline, with some 3500 pages completed in draft form and about 1700 in final, camera-ready copy.

As a tool for high-volume production, TEX has proven itself to us at Calma R&D. It *does* require quite a bit of familiarization and practice, but the results so far have been worth it.

* * * * * * * * * * *

TEX AT TEXAS A&M UNIVERSITY

Norman Naugle and Bart Childs

The best news about TEX at TAMU is the arrival of a QMS Lasergrafix 1200 printer and the ability to produce TEX82 *dvi* files successfully on a Data General MV8000. Although we have been running TEX using VAX/VMS systems, the lack of a reasonable output device has hampered our progress. Until now, our output devices have been Printronix and Trilog printers — suitable for overhead projectors. These printers use a driver (modified) and 200 bpi fonts obtained from Oregon Software.

Initially, the QMS printer will be used in the graphics mode to paint pages, but development of a driver is under way. The printer will have TEX82 fonts at 300bpi in ROM plus down-loadable fonts. It is anticipated that the printer will digest *dvi* files a page at a time, i.e., from *bop* to *eop*. This strategy should allow the printer to approach its 12 page per minute capacity. Hooks for embedded graphics are possible and are being considered.

Our plan is to offer TEX82 on all major computers on the TAMU campus. This includes VAX, Prime, Amdahl, DG, and Cyber. Available output devices will include Lasergrafix, Versatec, Trilog, Printronix, Xerox 9700, and Linotron 202. We are hoping for something like an Autologic APS-5 in 1984.

Although we can't say TEX correctly since we're from TeXAS, we can say **Thank you!** to everyone in the TEX community for their generous support to an isolated outpost.