Porting TEX to C

Klaus Lichtenwalder
Datapat GmbH München

There are already several C versions of TEX available that claim to have passed the trip test, namely CommonTEX, TurboTEX and the version for the Commodore Amiga. It is also said that parts of MicroTEX are written in C. We have also developed a portable version of TEX in C in a one year project, which now has been upgraded to the latest (as far as we know) Version 2.5. Our version of TEX passes the Trip test (naturally, otherwise we wouldn't dare tell you) and has been very easily ported to a number of different machines, as you will see later on.

The translation process

TEX is supposed to be portable by any means (if you get a running version of the BSD pc Pascal compiler). If you look at the TEX distribution sources, you will find that TEX has actually been written in a meta-language called `WEB`. If you finally get the necessary things up und running and have your TEX Tangled to Pascal source (which is supposed to show up at the end), you probably might stop trying to port TEX. But if you know something about Pascal and aren't worried about editing 1Mbyte files, as we did, just keep on working, or, better, have your Pascal compiler carry on. So, depending on your Pascal compiler, you could have a running version of TEX or, more probably if you are not on a BSD machine, you will start to look at the 1MByte of error messages you just got. So did we, and encountered a problem with our Pascal compiler (probably a cross-compiled version from a 8/16-bit machine) in handling reasonably-sized (definitely *not* large) arrays. A problem had already shown up in compiling Tangle and Weave, two tools you need for handling `WEB` files. After trying to fix that problem in Pascal without losing too much efficiency, we stopped relying on Pascal and thought about using something else.

Not too far away from Pascal, and maybe a reasonable choice on a UNIX system, we started to think about C. Yet another reason was a C compiler that never showed up with a bug or the like (even the optimizer wasn't buggy, something you may encounter on some 680X0-machine or other). So with this promising background, we took a close look at the some 23,000 lines of source code and stopped working on that problem soon after, because vi didn't seem well suited for this problem, since standard vi is limited to a 256K input file! Handling multiple files is not comfortable and what about viewing two files at the same time? We split the source into many smaller units, so as to drastically reduce compile time when making minor changes in just one procedure, and to have better response time in handling text units of a few K instead of a few hundreds. People invented a make facility, so why not use it? (God bless UNIX!)

At this time, we got the source of a fairly big version of EMACS and tried to port it to our machine (then a V.0 68000 machine). This decision turned out to have a major impact on the successful port of TEX, because of EMACS' sophisticated text handling commands and not easily surpassed size restrictions, and last, but by far not least, the modes that help you edit programming languages.

But anyway, having EMACS doesn't solve porting problems. First thing then was to invent some kind of Pascal beautifier. The rewrite process started afterwards. Line by line the Pascal source was replaced by the equivalent C code. There EMACS was of great help with global replace functions over all of its buffers, and with its keyboard macros. The latter were especially useful for transforming Pascal control structures to their C equivalents. We didn't intend to transform TEX into a hell of a C program (who would dare to change Knuth's intentions, or, worse, algorithms?), but to simulate Pascal restrictions as closely as possible so as not to lose portability (whatever degree of portability you expect). In the middle of the rewriting process (and after buying "TEX: The Program") we learned of some useful macros and `defines` we re-introduced into the Tangled source code for flexibility. So this prolonged step of rewriting into C took about half a year. The process of convincing the C compiler that the stuff he was reading was actually C did not take too long. There were quite a lot of misspellings and misconceptions and the like to get rid of, if you expected the program to do more than just print out the banner message "`This is TeX...`".

The major problem

One misconception, however, was of particular importance to the portability of the C version that slowly came into existence. In the `WEB` versions there are provisions for machines that do sign extensions, and for machines that don't. We ignored the preconditions, and, as always with a 50% chance, you get the wrong half. When we learned about the problem and also that this source wasn't intended for our machine, we went over the source code once again and spotted it with a handful of type casts.

The net effect is, if your C compiler knows how to handle casts (up to now, every C compiler we could test did), that this source runs on both types of machines.

Passing the Trip test

The important milestone after the banner is the Trip test. Needless to say, it didn't run at once. In fact, there were some very subtle bugs introduced while rewriting. It took another half a year to succeed with this test. One of the main problems lay in the input routine, where we didn't use Knuth's raw version, but the optimized version that happened to be in the change file (ever heard of a change file?). Needless to say, the algorithms and data structures in TeX are computer proof; that means, that if you have a compiler that deserves this name, you get this program running.

Sure enough, people learned about our project and asked for a port, if we ever got it running. Most of the time these people were more optimistic about a possible success than we were. But then we could make the (ultimate) test for portability.

The one thing we learned is, that TeX (whether in Pascal or in C or whatever) is not only a typesetting system, but also a compiler test system. There were some problems compilers introduced with the input of TeX, but if you wanted to demonstrate the bug to the computer or compiler distributor, you couldn't reproduce the error with a normal sized program. We encountered the fact that fixed array locations like `mem[32760]`, as happen to be used as kind of register in the typesetting processor TeX, will be translated to anything, but definitely not to the locations you would expect. Also you have to cope with the most tricky optimizers, which try to keep the program small enough by optimizing procedures away, or deleting the index in `z = mem[z]` before using it. But these problems were not too often encountered, and after tuning some I/O statements not for efficiency but for portability, we now have the following ports:

Cromemco V.0 and V.2
PCS Cadmus 32Bit UNIX Systems
ALTOS UNIX and XENIX Systems
Convex with 4.2 UNIX
AT&T 3b2 running V.0 and V.2/V.3
HP Series 9500 and 93XX under HP-UX
IBM RT under AIX

The only preconditions we pose are that we have a true 32-bit CPU (not an 80286) and we prefer UNIX or UNIX look-alikes, but we don't insist on this (as people insist on a VMS version).

Extensions

With this TeXinC version we started a cooperation with a German typesetter. In this project we designed an extended TeX program, which we call PhotoTeX, to cope with the possibilities available with phototypesetting machines, and made some adaptations for German respectively European environments. The PhotoTeX Program understands two additional keywords, `setsize` and `slantsize`, so that we are capable of handling dynamic fonts in the typesetting machine. Also we had to create metric files (tfm-files) for the fonts that are resident in phototypesetting machines, as there are machines that are not able to download fonts. Another hard problem was to find the right kernings, as typesetters need them.

An additional problem for the German environment is hyphenation of words with Umlaute (and other special characters you encounter in a European environment). At the moment, there are two solutions we know for hyphenation, both coming from the University in Bonn, Germany. One is to fool the hyphenation routine, while the other, and by far better, solution requires a minor change in the METAFONT description, recreating the fonts, and an addition in the dvi driver. We preferred the second approach for our German version of TeXinC.

TeX Adapted to CWEB

David Kennedy
Micro Publishing Systems, Inc.

This article announces TeX in CWEB, a new starting point for TeX ports. We have recently completed the translation of TeX to CWEB, a version of Don Knuth's WEB system of structured documentation, entirely rewritten in C, with many changes to take advantage of features found in C, but not in Pascal. (For a more complete description of CWEB refer to the TUGboat article: *WEB Adapted to C, Another Approach* by Silvio Levy, April, 1987).

Although this is a commercial venture, and the TeX translation is proprietary, we are offering a copy of the binary and/or source code for a reasonable license fee. We are also planning a fall 1988 commercial release of our fully TRIP-certified version of TeX for the PC and plan to release UNIX