> sets. There are two ways to make a TEX file machine dependent, both mentioned in the manual: (1) Name your files with local conventions that use something other than letters and digits. (2) Use non-printing ASCII characters in your source files.
>
> ...
>
> Some installations of TEX will accept TAB characters (and treat them as spaces); others will not. ... See *TEX: The Program*, section 23; the *xchr* and *xord* can be different on different computers.
>
> ...
>
> In other words, TEX is working just as we designed it. The users who expect identical behavior across machines have to abide by (1) and (2).

I'm still not happy, because there is still a problem. More and more TEX users are shipping files around to other sites, where they are to be TEXed to generate copy for publication, or read, perhaps changed, and shipped on somewhere else. Not all these locations will have identical hardware and software, and, given the opportunity, Murphy's Law always applies.

I don't have any good solution to this problem. The only (bad) solution I can think of is, if your system is one that handles this situation poorly, you should check every file for bare `^M`s and fix them up before submitting the file to TEX. I would welcome a discussion of this problem by the TEX implementors, since, although it appears to be a communications problem between disparate systems, it does seem to bend the "same input, same output" principle that we've come to expect TEX to follow.

Rules can be a deep subject

One of my correspondents sent me an interesting puzzle a while ago. In what he thought was a simple use of a "fill-in" rule in an `\halign`, he suddenly found that the rules varied in thickness. Here is the preamble he was using:

```
\halign{# \hfill
   &# \vrule width 12pc height .5pt \cr
... }
```

and this is what he got:

Name: ____________________
Address: ____________________
____
Telephone: ____________________
Best time to call ____________________

The rule on the next-to-last line has taken on the depth of the "p".

This is what he wanted:

Name: ____________________
Address: ____________________
____
Telephone: ____________________
Best time to call ____________________

I can see two possible solutions.

- Explicitly specify the depth; otherwise the "environment" depth will be used. In the "corrected" version above, `depth 0pt` was added.
- Use leaders instead of a `\vrule`: `\leaders\hrule\hskip 15pc` gives the same result as the `\vrule`.

Actually, my correspondent had tried an `\hrule` first, but got the nasty message that leaders were required in that context. Remembering that one must always nest opposites (`\vrule`s in horizontal mode, `\hrule`s in vertical), and being too lazy to look up how to use leaders, he simply switched to the `\vrule`. A reasonable approach. But there are a few things that one must remember not to take for granted.

Macros

German TEX, a Next Step

Peter Breitenlohner

1 The Present State

As reported by Joachim Lammarsch (TUGboat 8(1987)304) and described in detail by Hubert Partl (TUGboat 9(1988)70–72) the German TEX Users Group has agreed on a standard for a "Minimal Subset of German TEX Commands" at its 6th meeting in Münster (Germany) last October. This standard was, in fact, designed to a large extent by Hubert Partl and the present implementation in terms of TEX–macros is almost entirely his work.

The basic idea is old: make " an active character and define " as a macro with one argument such that the macro expansion yields whatever is necessary.

The present implementation has the advantage of high portability since TeX, TFM files, (German) hyphenation patterns and GF/PK/PXL files all remain unchanged. The only thing needed is a file GERMAN.TEX or GERMAN.STY which contains the necessary macro definitions. In particular " is defined to yield something like:

```
"s  →  \char25
"a  →  \accent\char127 a\allowhyphens
"c  →  \discretionary
         {k-}{}{c}\allowhyphens
"t  →  \discretionary
         {tt-}{}{t}\allowhyphens
"|  →  \discretionary
         {-}{}{\kern.03em}\allowhyphens
```

where o and u are treated like a (umlauts) and f, l, m, n and others are treated like t (special German \discretionarys). This solution has, however, several disadvantages. First the \accents and \discretionarys inhibit automatic hyphenation of the full word, only the parts before and (due to \allowhyphens) after the \accent will be subject to TeX's hyphenation algorithm. The "Umlaut accent" of the computer modern fonts does not yield a really satisfactory German umlaut, not even after some vertical shifting as applied in Partl's macro definitions; ideally there should be single characters, specifically designed for German umlauts, at some character position above 128 and accessed via ligatures. Finally in many areas (e.g., physics, mathematics, ...) we actually live in a bilingual environement. Many publications (like the one you are reading in this moment) are written in English, other publications and most letters are in German. Ideally something like Ferguson's multilingual TeX (see e.g., TUGboat 6(1985)57–58) should be available for English-German-French-Italian-...

In this article I want to discuss what can be done with modifying the TFM files and the hyphenation patterns, but nothing else. This still guarantees a reasonable amount of portability and is similar to the approach for "French in TeX" described by Alonzo Gariepy (TUGboat 9(1988)65–69). The basic strategy can already be found in Bernd Schulze's article in TUGboat 5(1984)103–104 and he also has created German hyphenation patterns with umlauts.

I hope some day there will be fonts with genuine German umlaut characters, but then it will need some more time until such fonts are available for all kinds of output devices. Moreover I have a strong suspicion that many of the existing device drivers do not handle the characters 128–255 correctly.

As far as multilingual TeX is concerned, it will be a long time until it will (perhaps) be available for all kinds of computers. To make multilingual TeX available as public domain software would certainly help this process. At present I see no point to invest any work in this direction.

2 The Second Step

Let me now discuss in detail how German TeX can make use of kerns and ligatures. These ligatures are not, as for the French accents, straightforward ones but they are mostly artificial "trick ligatures". In order to work they must use (abuse) two characters which are (at least in PLAIN TeX and in LaTeX) never accessed directly but only through control sequences. For the moment let us denote them by α and β.

2.1 The Macros

First Partl's macro definitions are modified to yield:

```
"a  →  aαa
"c  →  cββ
"t  →  ttα
"|  →  β
```

and similar for other characters. The α and β in these definitions are two suitable characters with category code 11 (letter) or 12 (other character). It is important that the above macro expansions involve nothing other than expansion of expandable tokens in order to use them in the hyphenation patterns. The macro definition may, however, start with \relax (or with LaTeX's \protect) provided we define \let\relax=\empty while reading the hyphenation patterns.

2.2 The Hyphenation Patterns

The first part of the hyphenation patterns is made by hand and contains:

```
a8α8a % never hyphenate inside "a
c8k % never hyphenate inside ck
8c8k9k8 % hyphenate ckk as ck-k
8c8β9β8 % hyphenate "ck as cβ-βk
8t8t9t8 % hyphenate ttt as tt-t
8t8t9α8 % hyphenate "tt as tt-αt
8t8z9z8 % hyphenate tzz as tz-z
f9β8 % hyphenate f"|l as f-βl
```

and similar for other characters. These patterns should in fact be \input from a separate file.

The remaining patterns are as usual and may contain "a, "o, "u and "s which will be converted into aαa and so on. One can in fact adapt Bern Schulze's file GHYPHENU.TEX for this purpose, but I hope that new and more efficient patterns will be available in the near future.

2.3 The TFM files

The ligatures and kerning in the TFM files (but nothing else) must be modified. First we add ligatures:

aα	→	¨						
cβ	→	k	kβ	→	c	βk	→	k
tα	→	α	αt	→	t			
βf	→	f	βi	→	i	βl	→	l

with the effect that:

"a	→	¨a		
"ck	→	ck	or	k-k
"tt	→	tt	or	tt-t
f"|l	→	fl	or	f-l

Note that the kerning between e.g., "a and the adjacent characters is the same as between a and the same adjacent characters. This will in general (although not in all cases) be just what is desired.

As a last step we introduce a small positive kern between f (or the ff ligature) and β as well as a negative kern between ¨ and either a or o or u in order to obtain e.g.,

"a	→	ä		
f"|l	→	fl	or	f-l

This kerning works nicely for "a and "o since the umlaut accent has the same width as a and o. The u is, however, somewhat wider, and the placement is somewhat different than what TEX's \accent would produce. In my opinion this is a tolerable price for the full automatic hyphenation achieved this way.

Without modifying pixel files the only alternative would be to convert "u into the sequence u \kern ¨\kern u such that the accent is placed as desired and the two u's are on exactly the same position. With DVItype's pixel rounding mechanism this leads, however, to the possibility that the two u's are off by one pixel.

Note that the mechanism described above works only for lower case umlaut characters. For the upper case ones one has to rely on TEX's \accent or something similar. This seems tolerable since the remaining part of the word (after the initial character) will still be hyphenated automatically.

These modifications of the TFM files will be done automatically by a suitable program which is under developement. The resulting files should certainly be named differently than the original ones. Deviating from the French solution I propose not to change the file name but rather the extension, e.g., from .tfm to .grtfm or .gfm. For all the non text fonts (such as math italic, math symbol and others) these .gfm files would be identical to their .tfm counterparts. Supplying INITEX with a modified POOL file (just replace the string .tfm by .gfm) results in the effect that INITEX reads .gfm files. This will also be true for TEX once the .fmt file produced by INITEX has been digested.

2.4 The Choice of α and β

The use of α and β in the trick ligatures for German hyphenation should not interfere with the normal use of these two characters. We propose to use the two character codes 0 and 1, normally accessed through the control sequences \Gamma and \Delta. Since both \Gamma and \Delta are defined as \mathchar they will never be part of a ligature nor will they be affected by (implicit) kerning. Therefore the definition of these two control sequences need not be modified; this was the reason to choose them instead of e.g., $ or #.

Finally we have to assign non vanishing \lccodes to the two characters α and β in order that TEX accepts them as part of hyphenatable words. When this scheme was designed, I was under the erroneous impression that the \lccode for α and β must be equal to the character code. This would have prevented the use of character codes 0 (vanishing \lccode) and 1 (a bug in TEX does not allow \lccode=1 in the hyphenation patterns) for α and β, instead one could have used character codes 2 (\Theta) and 3 (\Lambda). Meanwhile Professor Knuth has pointed out to me that the \lccodes need not and in fact should not be equal to the character codes. Instead they should be close to the character codes for the lower case letters a...z in order that the hyphenation patterns can be stored and retrieved efficiently, i.e., we can define:

```
\lccode0=95
\lccode1=96
```

3 Conclusion

The implementation of "German TEX" described here will be achieved with a moderate amount of work (both man power and CPU time). It has the advantage that the resulting DVI file can be processed with the standard computer modern pixel files and that almost all words can be automatically hyphenated. The disadvantage is clearly that the shape of the resulting umlaut characters is a not fully satisfactory compromise.

Once the whole program is completed experience will teach us the weak points of the present design and how to eliminate them.