Hidden treasures

At ConTEXt meetings we always find our moments to reflect on the interesting things that relate to TEX that we have run into. Among those we discussed were some of the historic treasures one can run into when one looks at source files. I will show examples from several domains in the ecosystem and we hereby invite the reader to come up with other interesting observations, not so much in order to criticize the fantastic open source efforts related to TEX, but just to indicate how decades of development and usage are reflected in the code base and usage, if only to make it part of the history of computing.

I start with the plain TEX format. At the top of that file we run into this:

```
% The following changes define internal codes as recommended
% in Appendix C of The TeXbook:

\mathcode`\^^@="2201 % \cdot
\mathcode`\^^A="3223 % \downarrow
\mathcode`\^^B="010B % \alpha
\mathcode`\^^C="010C % \beta
\mathcode`\^^D="225E % \land
...
\mathcode`\^^Y="3221 % \rightarrow
\mathcode`\^^Z="8000 % \ne
\mathcode`\^^[="2205 % \diamond
\mathcode`\^^\="3214 % \le
\mathcode`\^^]="3215 % \ge
\mathcode`\^^^="3211 % \equiv
\mathcode`\^^_="225F % \lor
```

This means that when you manage to key in one of these recommended character codes that in ASCII sits below the space slot, you will get some math symbol, given that you are in math mode. Now, if you also consider that the plain TEX format is pretty compact and that no bytes are wasted,[1] you might wonder what these lines do there. The answer is simple: there were keyboards out there that had these symbols. But, by the time TEX became popular, the dominance of the IBM keyboard let those memories fade away. This is just Don's personal touch I guess. Of course the question remains if the sources of TAOCP contain these characters.

1. Such definitions don't take additional space in the format file.

There is another interesting hack in the plain TEX file, one that actually, when I first looked at the file, didn't immediately made sense to me.

```
\font\preloaded=cmti9
\font\preloaded=cmti8
\font\preloaded=cmti7

\let\preloaded=\undefined
```

What happens here is that a bunch of fonts get defined and they all use the same name. Then eventually that name gets nilled. The reason that these definitions are there is that when TEX dumps a format file, the information that comes from those fonts is embedded to (dimensions, ligatures, kerns, parameters and math related) data. It is an indication that in those days it was more efficient to have them preloaded (that is why they use that name) than loading them at runtime. The fonts are loaded but you can only access them when you define them again! Of course nowadays that makes less sense, especially because storage is fast and operating systems do a nice job at caching files in memory so that successive runs have font files available already.

Talking of fonts, one of the things a new TEX user will notice and also one of the things users love to brag about is ligatures. If you run the `tftopl` program on a file like `cmr10.tfm` you will get a verbose representation of the font. Here are some lines:

```
(LABEL C f)  (LIG C i O 14) (LIG C f O 13) (LIG C l O 15)
(LABEL O 13) (LIG C i O 16) (LIG C l O 17)
(LABEL C `)  (LIG C ` C \)
(LABEL C ')  (LIG C ' C ")
(LABEL C -)  (LIG C - C {)
(LABEL C {)  (LIG C - C |)
(LABEL C !)  (LIG C ` C <)
(LABEL C ?)  (LIG C ` C >)
```

The `C` is followed by an ASCII representation and the `)` by the position in the font `O` (a number) or `C` (a character). So, consider the first two lines to be a puzzle: they define the fi, ff, fl ligatures as well as the ffi and ffl ones. Do you see how ligatures are chained?

But anyway, what do these other lines do there? It looks like ` `` ` becomes the character in the backslash slot and `''` the one in the double quote. Keep in mind that TEX treats the backslash special and when you want it, it will be taken from elsewhere. But still, these two ligatures look familiar: they point to slots that have the left and right double quotes.[2] They are not really ligatures but abuse the ligature mechanism to achieve a similar effect. The last four lines are the most interesting: these are ligatures that (probably) no TEX user ever uses or encounters. They are again something from the past. Also, changes are low that you mistakenly enter these sequences and the follow up Latin Modern fonts don't have them anyway.

Actually, if you look at the Metafontand MetaPost sources you can find lines like these (here we took from `mp.w` in the LuaTEX repository):

```
@ @<Put each...@>=
mp_primitive (mp, "=:", mp_lig_kern_token, 0);
@:=:_}{\.{=:} primitive@>;
mp_primitive (mp, "=:|", mp_lig_kern_token, 1);
@:=:/_}{\.{=:\char'174} primitive@>;
mp_primitive (mp, "=:|>", mp_lig_kern_token, 5);
@:=:/>_}{\.{=:\char'174>} primitive@>;
mp_primitive (mp, "|=:", mp_lig_kern_token, 2);
@:=:/_}{\.{\char'174=:} primitive@>;
mp_primitive (mp, "|=:>", mp_lig_kern_token, 6);
@:=:/>_}{\.{\char'174=:>} primitive@>;
mp_primitive (mp, "|=:|", mp_lig_kern_token, 3);
@:=:/_}{\.{\char'174=:\char'174} primitive@>;
mp_primitive (mp, "|=:|>", mp_lig_kern_token, 7);
@:=:/>_}{\.{\char'174=:\char'174>} primitive@>;
mp_primitive (mp, "|=:|>>", mp_lig_kern_token, 11);
@:=:/>_}{\.{\char'174=:\char'174>>} primitive@>;
```

I won't explain what happens there (as I would have to reread the relevant sections of TEX The Program) but the magic is in the special sequences: `=:` `=:|` `=:|>` `|=:` `|=:>` `|=:|` `|=:|>` `|=:|>>`. Similar sequences are used in some font related files. I bet that most MetaPost users never entered these as they relate to defining ligatures for fonts. Most users know that combining a `f` and `i` gives a `fi` but there are other ways to combine too. One can praise today's capabilities of OpenType ligature building but TEX was not stupid either! But these options were never really used and this treasure will stay hidden. Actually, to come back to a previous remark about abusing the ligature mechanism: OpenType fonts are just as sloppy as TEX with the quotes: there a ligature is just a name for a multiple-to-one mapping which is not always the same as a ligature.

But there are even more surprises with fonts. When Alan Braslau and I redid the bibliography subsystem of ConTEXt with help from Lua, I wrote a converter in that language. I actually did that the way I normally do: look at a file (in this case a bibTEX file) and write a parser from scratch. However, at some point we wondered how exactly strings got concatenated so I decided to locate the source and look at it there. When I scrolled down I noticed a peculiar section:

```
@^character set dependencies@>
@^system dependencies@>
Now we initialize the system-dependent |char_width| array,
for which |space| is the only |white_space| character given
a nonzero printing width.  The widths here are taken from
Stanford's June~'87 $cmr10$~font and represent hundredths
of a point (rounded), but since they're used only for
relative comparisons, the units have no meaning.

@d ss_width = 500 {character |@'31|'s width in the $cmr10$ font}
@d ae_width = 722 {character |@'32|'s width in the $cmr10$ font}
@d oe_width = 778 {character |@'33|'s width in the $cmr10$ font}
@d upper_ae_width = 903 {character |@'35|'s width in the $cmr10$
font}
@d upper_oe_width = 1014 {character |@'36|'s width in the $cmr10$
font}

@<Set initial values of key variables@>=
for i:=0 to @'177 do char_width[i] := 0;
@#
char_width[@'40] := 278;
char_width[@'41] := 278;
char_width[@'42] := 500;
char_width[@'43] := 833;
char_width[@'44] := 500;
char_width[@'45] := 833;
```

Do you see what happens here? There are hard coded font metrics in there! As far as I can tell, these are used in order to guess the width of the margin for references. Of course that won't work well in practice, simply because fonts differ. But given that the majority of documents that need references are using Computer Modern fonts, it actually might work well, especially with Plain TEX because that is also hardwired for 10pt fonts. Personally I'd go for a multipass analysis (or maybe would have had bibTEX produce a list of those labels for the purpose of analysis but for sure at that time any extra pass was costly in terms of performance). That code stays around of course. It makes for some nice deduction by historians in the future.

I bet that one can also find weird or unexpected code in ConTEXt, and definitely on the machines of TEX users all around the world. For instance, now that most people use utf8 all those encoding related hacks have become history. On the other hand, as history tends to cycle, bitmap symbolic fonts suddenly can look modern in a time when emoji are often bitmaps. We should guard our treasures.

Hans Hagen,
February 2020

2. ConTEXt never assumed this and encourages users to use the quotation macros. Those ``` ``quotes'' ``` look horrible in a source anyway.