Looking over the Debugger's Shoulders

When a computer bug has several causes and is hard to reproduce, I find it interesting to look over the shoulders of the debuggers and watch them at work. This is such a report. It is a summary of the debugging tour de force by Melissa O'Neill, Thanh The Han, and Jin-Hwan Cho (with help from Jonathan Kew and Karl Berry) which led to an understanding of the two causes of the notorious font cache bug.

The event began with the following post by Melissa in the pdftex mailing list:

> The enclosed patch fixes a longstanding bug in font subsetting
> for pdflatex that has existed since 1.20a-rc1 (March 2004). This
> bug has been causing problems for Mac users for some time
> because OS X does not like the subtly invalid fonts that pdflatex
> can output (but it may have affected others as well).
>
> The technical details and patch are below...
>
> The issue relates to CharString encryption for Type 1 fonts.
>
> In particular, the lenIV parameter of a font controls how much
> (encrypted) padding to put in front of the (encrypted) data.
> The default is four, so four bytes of encrypted padding are
> produced.
>
> In one part of the pdftex font-subsetting code (the part that
> writes out empty subroutines that have been dropped in the
> process of subsetting), pdftex incorrectly treats a padding
> of zero as meaning that it should abandon charstring encryption
> altogether. Not so. (FWIW, by undocumented convention, a
> padding of -1 does, in fact, mean no encryption.)
>
> By not encrypting data that is supposed to be encrypted, the
> subroutine ends up containing operator 0x1b, the hhcurveto
> operator, rather than 0x0b, return. This is an error in two ways,
> first, subroutines should return, and, second, the hhcurveto
> operator expects some arguments.
>
> Note that the subroutines that are affected by this bug are
> exactly those that are *never called*, so we might expect the
> bug to be harmless. It would appear, however, that Apple's font
> technology does process these "uncalled" subroutines
> and does not react well to their invalid content.

```
Thus, to have been affected by this bug, you need to

 a) Use a font where lenIV = 0, and

 b) Use a Mac (or anything else that needs uncalled
 subroutines to nevertheless be valid),

which explains why so many people have said, "I see
no problem; it works for me" in response to the anguished
cries of Mac users with PDFTeX-generated documents
everywhere.

FWIW, the buggy code was checked into the repository
for version 1.20a-rc1.  Prior to that point, this aspect of
font subsetting worked correctly.

The patch is, thankfully, trivial, changing (t1_lenIV > 0)
to  (t1_lenIV >= 0).

Fonts known to be affected by this bug include cm-super,
sanskrit,  and fonts processed with cfftot1 (i.e., OpenType
fonts used with  LaTeX), as well as many Adobe Type 1
fonts.
```

Melissa enclosed a patch file for pdftex, and gave a link to a patched version of pdftex on her web site. Almost immediately, the patch was accepted by Thanh The Han and then incorporated into LuaTeX by Martin Schroder. (LuaTeX has seen many changes since TeX Live 2008, so we thought it safest not to switch to the current version until TeX Live 2009.)

Karl Berry wrote a few people notifying them of the bug, in case their own code was affected. The most important recipient of this email was Jin-Hwan Cho, who immediately reported that his program dvipdfmx does not write out the sort of fonts affected by the bug. But he had also encountered the bug often, and concluded that the bug found by Melissa O'Neill was not the sole cause of the problem. Jin-Hwan sent the team a tex source which seemed to produce a defective pdf and used that example to support the belief that the bug was mainly an Apple bug, particularly a Leopard bug. In retrospect, this strong opinion helped galvanize the team into action.

In response, Melissa pointed out that when the bug first occurs, Apple's ATSServer reports font errors in the Macintosh console log. She wrote a script which opened a file or series of files hundreds of times while observing the console for errors. This test revealed that Jin-Hwan's example file did not trigger the bug, that bad files created by an unpatched

pdftex almost immediately triggered the bug, and that these same files created with a patched pdftex did not trigger the bug.

Using this script, Jin-Hwan tested many of his own files without finding a bad one. He asked Melissa if she had a bad file, and Melissa produced a file first made in 2005 which almost immediately produced the bug. The bad file used a subset of Adobe's TektonMM font, but was fixed by the pdftex patch.

At this point, evidence pointed to the pdftex bug as the sole cause of the problem. But Jin-Hwan's doubts continued, and that forced the rest of the team to continue working.

Thanh produced the next key piece of evidence. He extracted the TektonMM font from Melissa's bad file and produced two pdf files from this font. One contained the letters "i" and "p" only, and the other contained all other glyphs. Both pdf files were produced with a patched pdftex. But the file containing "i" and "p" produced errors with Melissa's test harness.

Cho and Melissa repeated this experiment. To a casual outsider like me, the testers were "thrashing around." The following message from Melissa is an indication of their progress:

> Here's some more data. Here are two versions of TektonMM
> containing just two characters. The first one (TektonYY) produces
> a PDF file that ATSServer hates, and the second one produces
> one that it is okay with.
>
> Interestingly, I made the second font by passing it through
> pstopdf and then extracting the font, so in one sense it still
> *is* the exact same font.

This message led to the following from Thanh:

> This is really interesting. I compared the two fonts and one
> thing that looks suspicious is that the "good" font doesn't
> make use of subr (apart from the first 4 ones). The call to
> subrs are replaced by the subr contents.

Melissa replied:

> The "good" one is more optimized. Apple's code subsetting
> code inlines any subroutine that is called only once.   I'm not
> sure that it's a meaningful difference. Possibly *any* chance
> would have stopped it from being "bad".

This produced an email from Jin-Hwan which I call the "breakthrough moment." The email began

> Here is my test to find why the mysterious things (found by

```
Han) happen.  Maybe I am the only one who does not need
to go bed.
```

This was followed by the following:

```
Now I explain why only i.pdf and comma.pdf have problems.
At first, see the file i.ps, 1444th line. Here the drawing
command for "i" ends with

372 callsubr

Now check the 372th subroutine, 1400th line.

dup 372 {
       11 5 div
       6 38 5 div
       41 5 div
       0 61 5 div
       rrcurveto
       closepath
       endchar
       }

Notice that it does not end with "return"! If I add one line
containing "return" after "endchar", the error message
disappears. I will explain later how to confirm.

Now, see the file comma.ps, 1574th line. Here you can
also see "424 callsubr". Check again the 1549th line,
the 424th subroutine.
```

```
dup 424 {
       -61 5 div
       22 5 div
       -31 5 div
       78 5 div
       vhcurveto
       closepath
       endchar
       }

Here it also ends with just "endchar" without "return".

Finally check p.ps. It's the most interesting one. The
1308th line is "266 callsubr", and you can find the
266th subroutine at 1039th line. It also end with
"endchar" without "return". However, another
subroutines follow, and the last subroutine ends
with "return".

As a matter of fact, I do not know exactly why this
does not cause error. (Question 1)

However, we know that in all three cases, those
subroutines end with "endchar" without "return".
So I translated ptkb8y.pfb to ptkb8y.ps using t1disasm
and then added return commands in the three
subroutines. After that made a new ptkb8y_n.pfb
using t1asm.

Now I can make PDF files which do not generate
any error with Melissa's catpdf.

It was a quite long trip. My conjecture for this notorious
bug is the absence of "return"  in subroutines. What
do you think?
```

At this point, Jonathan Kew wrote

```
Great work, ChoF (and all who have been chasing this issue)....
looks like you are onto something important.

Note that the Adobe specification for Type 1
explicitly says that
```

> "A charstring subroutine must end with
> the return command."
>
> (Adobe Type 1 Font Format, version 1.1, bottom of p.67).
>
> So I think whatever tool generated this font is incorrect.
>
> > Finally check p.ps. It's the most interesting one. The
> > 1308th line is "266 callsubr", and you can find the 266th
> > subroutine at 1039th line. It also end with "endchar" without
> > "return". However, another subroutines follow, and the last
> > subroutine ends with "return".
> >
> > As a matter of fact, I do not know exactly why this does
> > not cause error. (Question 1)
>
> I'd guess that the entire subroutines array is treated as
> a single large buffer of opcodes, with each subroutine
> entry being an index into this array. If a subroutine is
> lacking its "return", interpretation will simply continue into the
> following subroutine; in the case of p.ps, the next is an empty
> subroutine that contains nothing but "return", and so calling
> subr 266 ends up returning safely.
>
> In the case of i.ps and comma.ps, the subr in question is the
> last one in the array, and so execution falls off the end of the
> subr array and tries to interpret whatever happens to follow....
> leading to the problem.
>
> > It was a quite long trip. My conjecture for this notorious bug
> > is the absence of "return" in subroutines. What do you think?
>
> I think that's highly likely.
>
> I don't know the actual origin of the bad subroutines; possibly
> some old font-creation tool. But what would be really useful would
> be for pdftex (and dvipdfmx, if an equivalent situation applies when
> writing the font as CFF) to check while embedding the font that
> each subroutine ends with "return", and add it if necessary.

Shortly after that, Jonathan wrote

> From the Type 2 Charstrings spec, it seems that subrs must
> end with either "return" or "endchar".

and then

> Another interesting question: what is the original source of
> the bad subroutines? Is this error present in the original
> TektonMM font from Adobe, or is it introduced by some other
> tool such as mmpfb that has processed the font data? Are there
> other fonts that suffer from this same error, and what is their source?

Thanh then wrote to Jin-Hwan

> I think you got it, bravo! This is certainly a bug of the font,
> since the t1 spec said that a subr must end with the
> "return" command.
>
> so we got 2 things confirmed now that makes leopard "unhappy":
>
> - the bug of pdftex caught by Melissa
> - the absence of the "return" command in some buggy fonts

and shortly after that, Melissa wrote:

> Excellent detective work, people. Awesome team effort!
>
> I can confirm that the original TektonMM font does not have the
> "subroutine does not return bug", but after processing with mmpfb,
> subroutines 266, 269, 304, 319, 321, 330, 352, 359, 372, 391, 392,
> 393, 394, and 424 do not return (ending with endchar). Thus,
> some blame is due to mmpfb.

At this point I'll stop quoting email messages, but summarize the remaining debugging.

Melissan and Thanh almost simultaneously then tested all pfb fonts in TeX Live to find fonts with a subr with missing "return." They found over 100 examples of such fonts, and discovered that most or all had been created with the TeXtrace utility and mmpfb. The author of mmpfb was then contacted. He discovered that mmpfb added the required "return" statements, but a later section of the code stripped them out. A cause of this confusion was probably that Type 1 fonts require the return statement, but type 2 CharStrings can end with either endchar or return.

The broken fonts with missing RETURN subroutines are now being fixed in TeX Live. Note that tlmgr and TeX Live Utility can upgrade fonts, so these repairs will filter out to users. In addition, Thanh patched pdftex so it repairs missing RETURN instructions in fonts it embeds in pdf files. This patch has also been applied to dvips. Thus the two key problems discovered by this debugging are repaired in the new pdftex and dvips.

But there is slightly more to the story. At the end, Melissa tested all fonts in TeX Live with t1lint, which can determine other incorrect features in fonts. The t1lint program is quite strict and can report things that aren't problems in practice. This test revealed a large number of additional fonts in TeX Live with problems. A number of fonts have missing BlueValues for their private dictionary. Melissa adds that fontforge's fontlint also complains about the quality of Blue Values in various Adobe fonts. Melissa also found a number of bad pdf files in TeX Live.

Seeing this message, Karl Berry wrote

> Yikes, yikes, yikes. In general, those pdf's are generated by each individual author when they are uploaded to CTAN. I don't want to build them myself in TL, there are too many strange environment issues that could cause the output to be silently wrong.
>
> As for the fonts, most of them are highly unlikely to have any active author who wants to look at any issues, but we can ask. I could imagine running t1lint on new uploads at least ...
>
> Anyway, I'll talk with the CTAN guys about all this too. We're proceeding with the subr font fixes for sure (based on your fixedfonts.tgz).

We summarize this development:

Two bugs contributed to the problem. First, pdftex, dvips, luatex, and metapost were not correctly subsetting certain fonts. And second, a large number of fonts contained subroutines which did not conclude with a RETURN command. Many of these were created with the utility mmpfb, which has been patched.

The fixes in pdftex and dvips correct both problems.

Finally, there may be problems in other fonts, but we do not have evidence that these problems are involved in the bad cache bug. These problems will gradually be fixed over the coming months.

Dick Koch