Fonts and PDF via pdfTeX

Tom Kacvinsky

EuroTeX 2001, Kerkrade, The Netherlands

Overview

- Overview of PDF file format
- Overview of font objects in PDF files
- How pdfTEX treats fonts
- Debugging font problems in PDF files
- Conclusion

Overview of PDF file format

- Described in chapter 3 of PDF Reference Manual, second edition.
- Basic building blocks are called **objects**
- Objects are built out of boolean values, numbers (real and integer), strings, names, arrays, dictionaries, streams, and the null object.

Boolean objects

- **true** or **false**

Number objects

- Integers: one or more decimal digits optionally preceded by a sign
- Real numbers: one or more decimal digits, with leading, embedded or trailing period (decimal point), with an optional sign
- Number objects may not be specified with different radixes

String objects

- Sequence of unsigned bytes.
- There are limits on how a string can be.
- Strings are delimited by (and) (binary encoded strings) or < and > (hexadecimal encoded strings)
- The escape mechanism in a string is \. Used to escape parentheses, amongst other things.
- If a binary encoded string contains matching (and), then the parentheses to not need to be escaped.
- If a binary encoded string contains an unmatched (or), it must be escaped, as follows: \(and \)
- \ is used to include other character codes in a binary encoded string. Examples include \ itself (\\), newlines (\n), and nonprinting ASCII characters
- \ is used to introduce octal representation of character codes. An example is \001, the octal representation of TAB.

String objects (continued)

- \ is used for line
- For hexadecimal encoded strings, white space characters are ignored. White space characters are space, tab, carriage return, line feed, and formfeed.
- In a hexadecimal encoded string, two hexadecimal digits form one byte; if the count of hexadecimal digits is not even the hexadecimal encoded string is padded with one 0 at the end.

Example String objects

```
(Hi!\243 \(here we begin\
      a parenthetical)

(Hi!\243 \(here we begin      a parenthetical)

<4869 2109486F7720617265207 96F753F205C2868657>

<48692109486F7720617265207 96F753F205C28686570>
```

Name Objects

- Name objects start with a /. The slash is not part of the name.
- Name objects are unique
- Names cannot include delimiter or white space characters
- Names are case sensitive
- / by itself is a legal name, but I don't recommend using it!
- Example: /FooBarBletch

Array Objects

- Delimited by [and]
- Contains other objects (names, arrays, dictionaries, and even other arrays)
- Again, there are implementation limits for arrays.
- Example: [42 (Barfulicious) /TomIsWeird]

Dictionary Objects

- Delimited by << and >>
- Contains key/value pairs. Keys are name objects, and values are any kind of object, including other dictionaries
- If a key's value is **null**, the that key is treated as being absent
- Dictionaries are the main building blocks of PDF files
- Dictionaries in PDF file customarily have a /Type key with a value that determines what kind of dictionary it is, and sometimes a /Subtype for further identification of the dictionary's purpose. The object used for both /Type and /Subtype is a name object.
- Example: << /Conference << /EuroTeX (2001) >> /Foo 12 >>
- Weirdness: The specification states that if a dictionary contains two entries with the same key, the value is undefined. However, evidence points to the fact that the last occurence of the key is the key whose value is taken. I believe that Martin Schröder reported this on the pdfTEX list.
- streams can be external to a PDF file; we will ignore this case.

Direct vs. Indirect objects

- An indirect object is an object that is labeled for later reference. The label consists of a positive number (the object number) and a nonnegative "revision" number. The PDF reference calls this latter number a generation number, but I think revision is a better name.
- The label is the object and revision numbers followed by **obj**. The object data then follows, and the data is followed by **endobj**.
- Nonlabled objects are direct objects.
- Later on, we will see that some PDF data structures must use indirect objects. They do so by using indirect references (their wording; I would use reference). This is done by using the object number and revision number followed by **R**. For example, if the object number is 2, and the revision number is 0, then an indirect reference for object 2 is "2 0 R".

Stream Objects

- Used to get more data than a string object, for unlike string objects, stream objects have no implementation limits (at least in theory).
- Delimited by **stream** and **endstream**
- stream objects **must** be indirect objects.
- The **stream** keyword must be followed by either `\r\n` or by `\n`, but not by `\r`. There are valid reasons for this, as explained in the PDF Reference Manual.
- stream objects can be compressed and encoded. The compression and encoding information is specified using a dictionary object.

General structure of stream objects

```
dictionary
stream(\r\n|\n)
<data>
endstream
```

Some keys used in stream dictionaries

- **Length**: (Required) An integer value that is length of the data, not counting any end of line tokens before the endstream keyword.
- **Filter**: (Optional) name or array of names giving decompression and decoding information for data in stream.
- **DecodeParams**: (Optional) dictionary or array giving information for decompression/decoding filters specified in **Filter**.
- There are more dictionary items, but these are not germane to streams contained within a given PDF file.
- For our purposes, the most common decompression filter is **FlateDecode**. The data that is decoded is thus FlateEncoded, which is the same data compression method as described in IETF RFC 1950. Or, more commonly, the gzip compression method.

An indirect object

```
12 0 obj
<< /Filter /ASCIIHexDecode /Length 44 >>
stream
48692109486F7720617265520796F753F205C28686570>
endstream
endobj
```

File layout

- Header
- Body
- Cross reference (**xref**) table
- File trailer
- For this talk, the salient file sections are the body, xref table, and the trailer.

Body section

- The body section is a sequence of indirect objects. Some objects are used to define "global" data, and some objects are used to describe page data.

The **xref** table

- The main idea behind the xref table is quick lookups of indirect objects.
- Each entry of the xref table is offsets from the beginning of the file to ann indirect object.
- There may be more than one cross reference table. These are referred to as sections. There is one section to begin with; each modification of the file may result in an additional xref table being added to the file.
- Each xref table itself might have more than one section.
- For more details on the cross reference table, see the PDF Reference Manual, second edition (or variants thereof), sections 3.4.3, 3.4.5, and G6.

Example xref table (taken from PDF Reference, Example 3.5)

```
xref
0 1
0000000000 65535 f\n
3 1
0000025325 00000 n\n
23 2
0000025518 00002 n\n
0000025635 00000 n\n
30 1
0000025777 00000 n\n
```

File trailer

```
trailer
<dictionary>
startxref
<byte offset to last xref section>
%%EOF
```

Discussion of file trailer

- There can be more than one xref table. This is accounted for in the dictionary after the trailer keyword. The particular entry is the **Prev** key, and its entry is an offset to the previous cross reference section.
- The **Root** dictionary is an indirect reference to an object that contains the **Document** catalog.

The **Document** catalog

- For this talk, the entry in the document catalog of interest is the **Pages** dictionary, and indirect reference to the root page in the page tree.
- There are other entries in the **Document** that contain information for bookmarks, etc...

The **Pages** object

- For this talk, the entry in the document catalog of interest is the **Pages** dictionary, and indirect reference to the root page in the page tree.

The page tree and **Pages** indirect objects.

- Acrobat Distiller makes the page tree a balanced tree, for quick lookups of particular pages. Think of binary trees and the binary search algorithm.
- The nodes in the page tree are called **Kids**. Each node in the page tree (except the root node) has a **Parent** node.
- The leaf nodes in the tree are the indirect objects that contain page layout information. These leafs are called **Page**s, and are dictionary objects.

The **Page** indirect object.

- Key entries are **Resources** and **Contents** indirect objects.
- The **Resources** indirect object is a dictionary that contains information about resources are used on that page.
- The **Contents** indirect object is a stream or array of streams (remember that streams are indirect objects) that contains the contents of the page (the page description).

Resource dictionary entries

- The **ProcSet** array. This array contains names of procedure sets used by the page. These procedure sets are for text operators, general pdf operators, or image operators.
- The **Font** dictionary. The keys are names used by the text operators and whose values are indirect references to **Font** objects.
- The **XObject** dictionary. The keys are named resources and whose values are indirect references to objects used by the page. For the XObject resources, it helps to think of graphics (bitmap or vector) that are used by the page.

Wow! It is past time for an example.

Fonts!

Supported font formats

- Type 0 (composite fonts)
- Type 1 (both regular Type 1 fonts and MM Type 1 fonts). In practice, only regular Type 1 fonts are used. If an MM font is used, it is already interpolated (that is, the interpolated MM font is a regular Type 1 font).
- Type 3
- True Type
- CID keyed fonts. The supported formats for CID keyed fonts are CIDFontType0 (Type 1 outlines) and CIDFontType2 (True Type outlines)

Objects related to fonts

- **Font** dictionary object. This is the top level font object. It contains encoding information and some metric information (namely, glyph widths).
- **FontDescriptor** dictionary object. This object is used by the top level **Font** object for certain fonts. It contains information about the font (more metric information, glyph complement, etc...)
- **FontFile**, **FontFile2**, and **FontFile3** stream objects. These stream objects are used to embed fonts in the PDF file. This is regardless of whether the font is fully embedded or is subsetted.

We will restrict our discussion to Type 1 fonts.

Type 1 **Font** objects

- **Type** Required; Name object; always **Font**
- **Subtype** Required; Name object; always **Type1**
- **BaseFont** Required; Name object that gives PS name of font. Used when printing PDF file to a PS device. Subsetted fonts have a **BaseFont** with a six character prefix, each character of said prefix is from the letters A-Z, followed by a +, followed by the "usual" name of the font. An example is **/AGFTYW+CMR10**.
- **FirstChar** Required; first character code defined in **Widths** array
- **LastChar** Required; last character code defined in **Widths** array
- **Widths** Required; array that gives glyph widths
- **FontDescriptor** Required; must be an indirect reference to a dictionary object.
- **Encoding** Optional; Can be a name (for a predefined encoding) or a dictionary that describes the encoding of the font.
- **ToUnicode** Optional; A stream object that maps character codes to Unicode values; useful for cut and paste and text searches.

Required entries in Type 1 **FontDescriptor** objects

- **Type** Name object; always **FontDescriptor**
- **Ascent** Number; maximum height of glyph above baseline, excepting accented glyphs.
- **CapHeight** Number; maximum height of flat capital letters
- **Flags** Number; bitfield describing properties of the font (italic, roman, symbol, smallcap, etc...)
- **FontBBox** Array of numbers; Numbers in array are described in glyph coordinates (usually 1000 units per EM). The **FontBBox** describes the smallest rectangle that encloses all glyphs in the font.
- **FontName** Name; must match the **BaseFont** name as given in the **Font** object.
- **ItalicAngle** Number; describes angle of dominant "vertical" strokes of glyphs in the font; measured in degrees counterclockwise from the vertical
- **StemV** Number; width of dominant vertical stems in the font.

Some useful optional entries in Type 1 **FontDescriptor** objects

- **FontFile** stream object; used for embedding of Type 1 "PFA" fonts
- **FontFile3** stream object; used for embedding of Type 1 "CFF" fonts
- **CharSet** string; list of glyphs in a subsetted font. An example is (/a/d/e/ff/i/l/n/r/t).

Example of a **FontFile** stream object

```
 27 0 obj
 <<
   /Length1 1521
   /Length2 8099
   /Length3 532
   /Length 8981
   /Filter /FlateDecode
 >>
stream
<8981 bytes deleted>
endstream
endobj
```

Example of a **FontFile3** stream object

```
8 0 obj
<<
/Filter [/ASCII85Decode /FlateDecode]
/Length 482
/Subtype /Type1C
>>
stream
<482 bytes deleted>
endstream
endobj
```