# 说明

这是对文件**file01\_text\_image\_table.pdf**的分析，大量参考了<https://blog.csdn.net/steve_cui>的文章

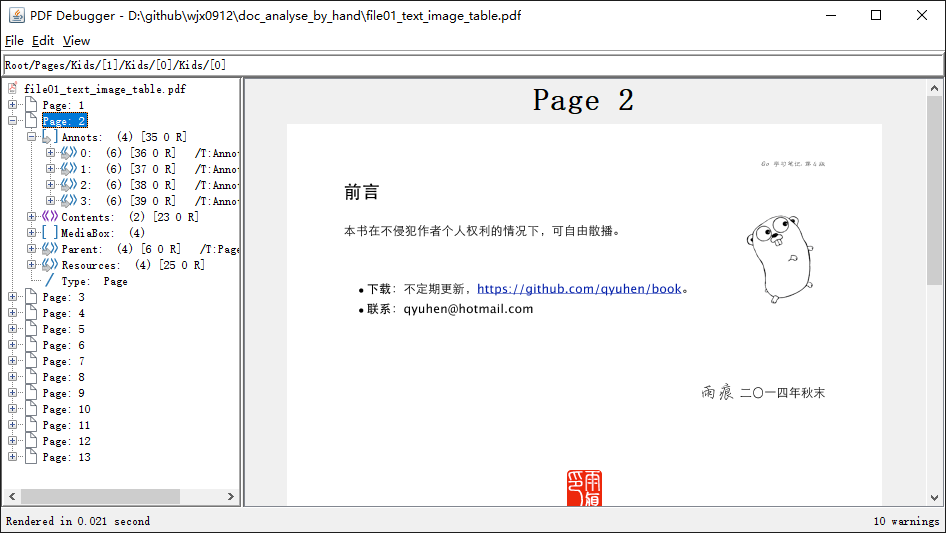
使用的工具有：

|  |  |
| --- | --- |
| notepad++ |  |
| IDM uestudio |  |
| apache的pdfbox的debugger-app-2.\*.\*.jar |  |
| CharsetAssistant.exe github上的开源项目 |  |

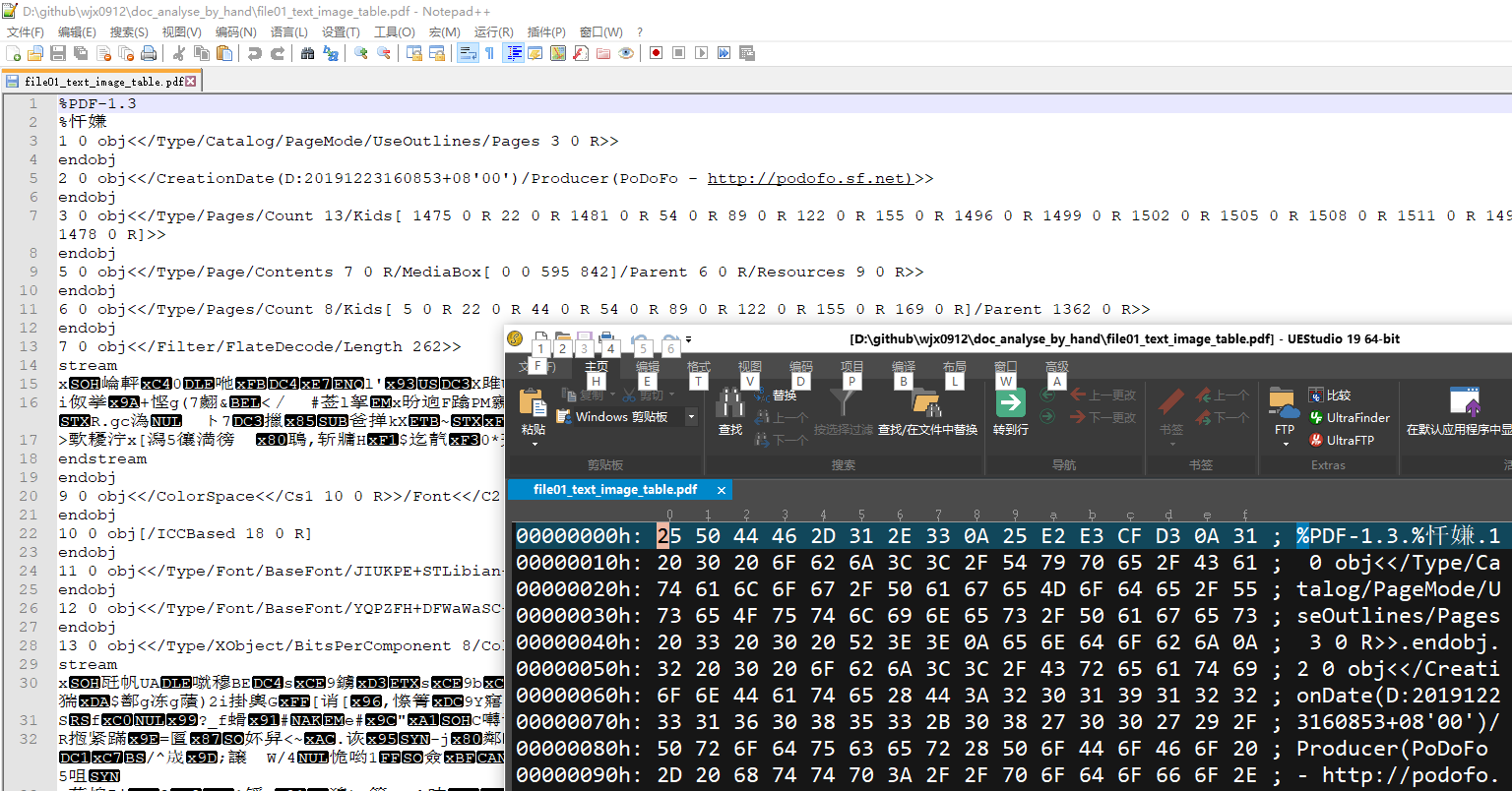
这个文件用adobe reader或者福昕都能正常打开（后者不能跳转索引）

先看下文件结构

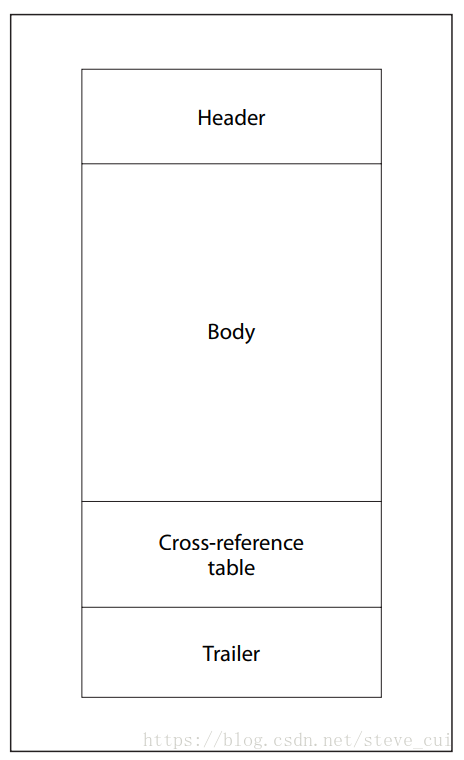
D:\>java -jar D:\github\apache\debugger-app-2.0.18.jar \

D:\github\wjx0912\doc\_analyse\_by\_hand\file01\_text\_image\_table.pdf

编辑器打开：



一个简单的pdf文件格式结构如下：（图来自：<https://blog.csdn.net/steve_cui/article/details/81948486>）



当然，这是最简单的。ref table和trailer可能有多个。

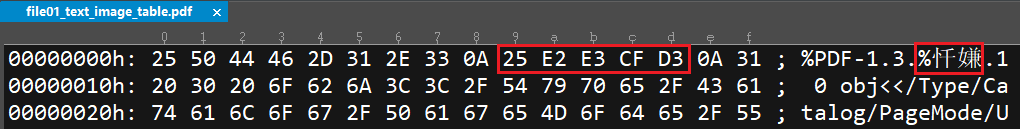
# 第1行

|  |
| --- |
| %PDF-1.3 |

1993年adobe发布第一个版本1.0，之后adobe将pdf格式捐赠给ISO后，于2009发布了1.7版本（这个规范就是PDF32000\_2008.pdf，以后会经常看到），之后的pdf版本几乎没有更新。

# 第2行

|  |
| --- |
| %忏嫌 |



这4个字节：E2 E3 CF D3

打开PDF32000\_2008.pdf，第7.2章，这4个二进制可以是任意数字，最好不是可见字符，这样其它软件如ftp就认为它是二进制文件而不是文本文件。

虽然这四个字节可以是任意值，但大家在创建pdf时都用这个值。

第1,2行也就是上面图中的header

# 第3,4行

|  |
| --- |
| 1 0 obj<</Type/Catalog/PageMode/UseOutlines/Pages 3 0 R>>  endobj |

从第3行到第11434行，一直都是这种结构，这些行也就是上面图中的body

# 第11435-12935行

|  |
| --- |
| xref  0 4  0000000000 65535 f  0000000015 00000 n  0000000080 00000 n  0000000177 00000 n  5 3  0000000377 00000 n  0000000473 00000 n  0000000592 00000 n  ……  ……  ……  0001190217 00000 n  0001195105 00000 n  0001195302 00000 n |

前面说到body定义了所有的obj对象，这里xref是obj对象的交叉引用表：用来保存各个间接对象在文件中的起始地址

上面第二行：0 4， 表示序号从0开始，有4个对象。对象0的起始地址是0000000000，版本号是65535，如果是保留对象版本一般是0，它最后有个属性f表示这个对象需要删除（f表示free）。

第六行0000000177 是对象3的起始地址，它最后有个属性n表示这个对象需要保留（n表示normal）。

解析的时候注意：每行对象描述一般是20字节，但我在某些pdf文件看到19字节的行。

# 第12936，12937行

|  |
| --- |
| trailer  <</ID[<BFA388FCACEC43E17877E10E34688547><BFA388FCACEC43E17877E10E34688547>]/Info 2 0 R/Root 1 0 R/Size 1513>> |

trailer是xref的摘要，表示里面有多少个obj，根对象（Root），加密对象（Encrypt），文档信息对象（Info）等。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 值 | 说明 |
| /ID | array | [<BFA388FCACEC43E17877E10E34688547>  <BFA388FCACEC43E17877E10E34688547>] | 分别表示旧id和新id |
| /Info | dictionary | 2 0 R | 文件摘要资讯的obj id |
| /Root | dictionary | 1 0 R | 【必选】catalog的obj id |
| /Size | integer | 1513 | 【必选】obj数量 |
| /Prev | integer |  | 指向下一个xref的trailer |
| /Encrypt | dictionary |  | 解密用的 |
|  |  |  |  |

说明：网上查的资料说明Prev指向上一个xref的地址。但实际分析一个pdf文件，发现指向的是trailer地址，trailer后面紧接着的startxref再执行xref地址。

# 第12938，12939行

|  |
| --- |
| startxref  1195409 |

第一个xref位置（如果有多个xref，每个紧跟xref的trailer里面可能有Prev字段）

# 第12940，12941行

|  |
| --- |
| %%EOF |

文件结束标记

# pdf文件格式简述

PDF基本显示单元包括：文字，图片，矢量图

PDF扩展单元包括：水印，电子署名，注释，表单，多媒体，3D

PDF动作单元：书签，超链接

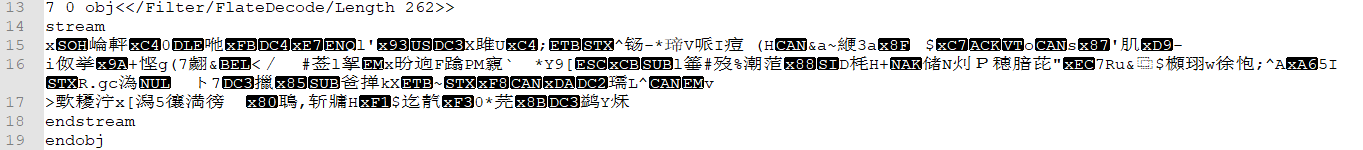
由于可以内嵌字体，pdf有良好的一致性；另外pdf文件普遍不易修改

PDF文件是由对象集合组成的，包括：boolean（布尔型），numberic（数值型），string（字符串型），name（名字型），array（数组型），dictionary（字典型），stream（数据流型），null（空类型）， indirect（间接型）

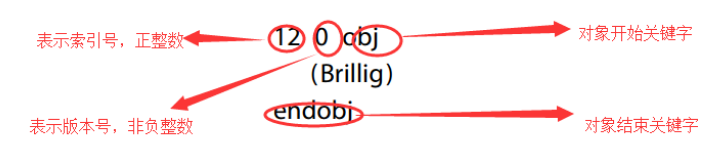
上面第3行，第12937行，用双尖括号**<< >>** ，就是典型的dictinoary类型；

文件中的第14行是stream类型；

indirect是最重要的类型：body里面的每个对象都是一个indirect类型，例如：



格式如下：



12是对象id，一般是递增（只要不重复即可） ；0是版本号；

# Filter

在PDF中，为了让文件变的更小，通常的做法是，将stream对象进行压缩，因为stream对象的数据块比较大，所以，是重点关注的地方。

从PDF1.5开始往后的版本，支持了Object stream，可以把多个对象（非stream对象）放到同一个stream对象中，并进行压缩，达到减少文件大小的效果。

Filter大部分的类型是压缩，当然有一个是特殊的类型”Crypt“，这个类型是对该stream进行单独加密。

# 关于PDF/A

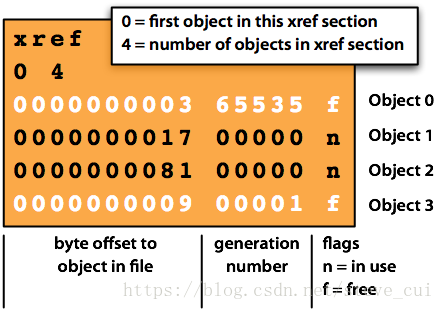
通常pdf在各个平台（或者打印机），显示效果是一样的。但是不排除一些特殊情况，例如引用了（非内嵌）某操作系统的字体，这个字体当前比较流行，但是多年后大多数os不再用这种字体，那么显示效果和最初可能存在差异。PDF/A将所有信息都归档到文件里面：通过嵌入在文档自身内部显示该文档的信息（内容、颜色、字体、图像，等等），存储这种自我持续。

另外还有个PDF/X，也是差不多的意思，但是只有专用场合才用。

# Cross-Reference Table 交叉引用表

PDF交叉参考表是PDF文件的重要部分。该表保存了所有简介对象在PDF文件中物理偏移地址。该表在文件中可以存在单个，也可以存在多个。多个交叉引用表通常出现在两个情况：一、增量保存，二、线性化。

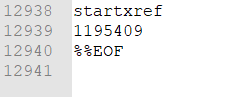
通常，PDF交叉引用表将具有以下形式：



交叉引用表以单词“xref”开头。图中，“0 4”表示对象号为0开始的连续4个对象，对象号为0和3的简介对象不存在，对象号为1和2的对象使用中。对象号为1的简介对象的起始位置是17，版本号是0.

一个文件中出现多个交叉引用表时，可能出现同一个间接对象存在不同的引用表中，这时，要以出现在文件最后位置的那个为准，前面的忽略，这种情况，通常是由于修改了PDF文件，导致其中的一个或多个对象发生了变化，PDF生成器根据输出要求，进行增加输出，只输出修改的对象，然后在文件末尾加上更新的交叉引用表。

那么交叉引用表的其实位置在什么地方呢？其实很简单，在文件的末尾保存有以下内容：



“1195409”就是交叉引用表的偏移位置。

当存在多个交叉引用表时，通常在交叉引用表之后的“trailer”字典中会保存“/Prev”，该key对应的值就是上一个交叉引用表的位置。

apache/pdfbox中解析xref的代码在：

FDFParser::initialParse

FDFParser::getStartxrefOffset

COSParser::parseXref

# pdf文件破损

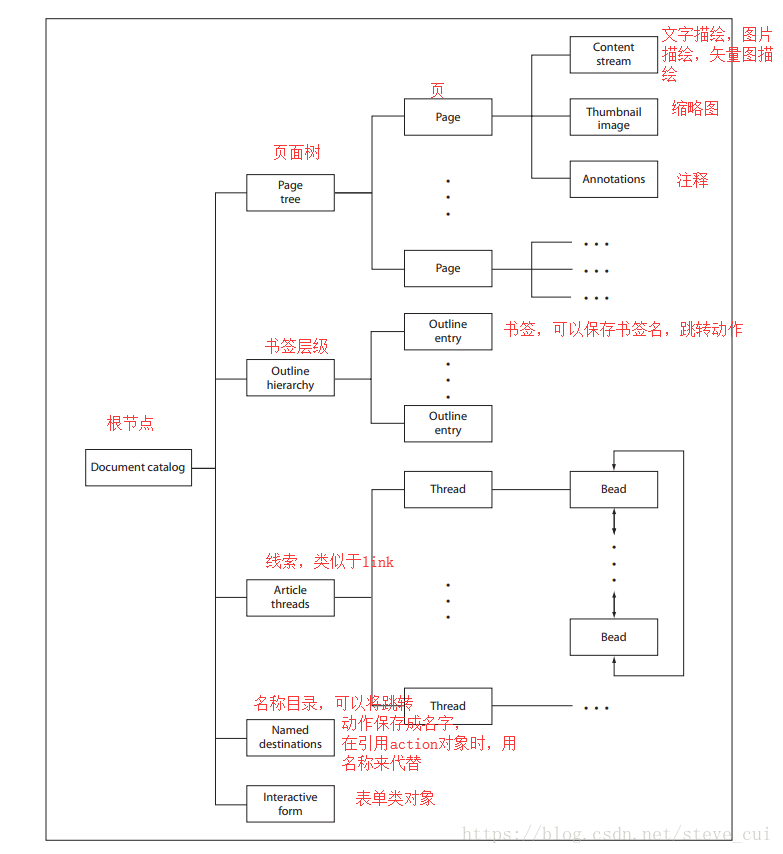
通常有2种情况：

1.obj破损

2.xref破损

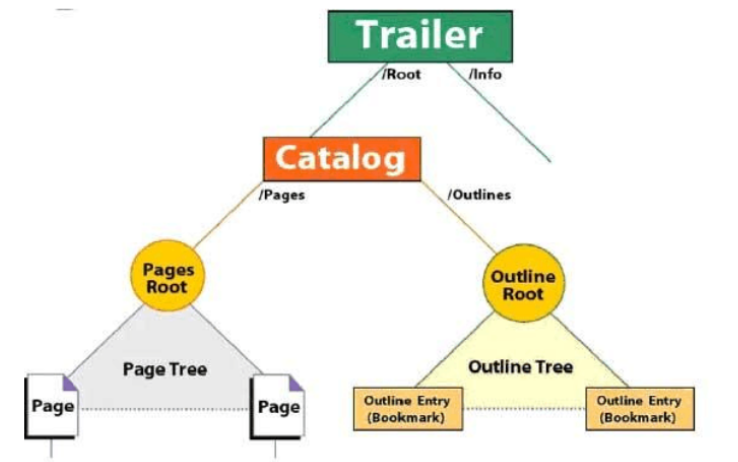
处理方法通常是重建xref表，参考：<https://blog.csdn.net/steve_cui/article/details/82185452>

# 完整文档结构图



图来自：<https://blog.csdn.net/steve_cui/article/details/81948486>

或者：



# 增量更新

<https://blog.csdn.net/steve_cui/article/details/82251702>

# 加密解密

<https://blog.csdn.net/steve_cui/article/details/82257603>

# Catalog文档目录及page tree页面树



把上面2行改成如下格式好理解一些：

|  |
| --- |
| 1 0 obj  <<  /Type /Catalog  /PageMode /UseOutlines  /Pages 3 0 R  >>  endobj |

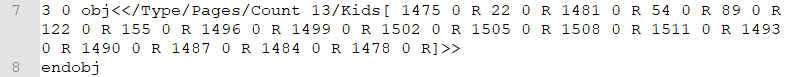
PageMode表示打开pdf时的操作为UseOutlines（书签），也就是指示编辑器打开pdf时打开书签窗口。

Pages 为3 0 R表示对象3里面包含pages信息。





trailer的字典里面的/Info使用了对象2



同样，这个obj可以格式化为：

|  |
| --- |
| 3 0 obj  <<  /Type /Pages  /Count 13  /Kids[ 1475 0 R 22 0 R 1481 0 R 54 0 R 89 0 R 122 0 R 155 0 R 1496 0 R 1499 0 R 1502 0 R 1505 0 R 1508 0 R 1511 0 R 1493 0 R 1490 0 R 1487 0 R 1484 0 R 1478 0 R]  >>  endobj |

pdf文档本身只有13页，但是/Kids页面信息里面有18个obj引用，这18个obj的id是：

|  |  |
| --- | --- |
| 1475 | 1502 |
| 22 | 1505 |
| 1481 | 1508 |
| 54 | 1511 |
| 89 | 1493 |
| 122 | 1490 |
| 155 | 1487 |
| 1496 | 1484 |
| 1499 | 1478 |

这些page内容如下（已格式化）

|  |
| --- |
| **1475** 0 obj  <<  /Type /Page  /**Contents** 1476 0 R  /MediaBox[ 0.000000 0.000000 595.000000 842.000000]  /Parent 3 0 R  /Resources<</ProcSet[/PDF/Text/ImageB/ImageC/ImageI]/XObject<</XOb1474 1474 0 R>>>>  >>  endobj |
| **22** 0 obj  <<  /Type /Page  /**Annots** 35 0 R  /**Contents** 23 0 R  /MediaBox[ 0 0 595 842]  /Parent 6 0 R  /Resources 25 0  R>>  endobj |
| **1481** 0 obj  <<  /Type /Page  /**Contents** 1482 0 R  /MediaBox[ 0.000000 0.000000 595.000000 842.000000]  /Parent 3 0 R  /Resources<</ProcSet[/PDF/Text/ImageB/ImageC/ImageI]/XObject<</XOb1480 1480 0 R>>>>  >>  endobj |
| **54** 0 obj<</Type/Page/**Annots** 61 0 R/**Contents** 55 0 R/MediaBox[ 0 0 595 842]/Parent 6 0 R/Resources 57 0 R>>  endobj |
| **89** 0 obj<</Type/Page/**Annots** 93 0 R/**Contents** 90 0 R/MediaBox[ 0 0 595 842]/Parent 6 0 R/Resources 92 0 R>>  endobj |
| **122** 0 obj<</Type/Page/**Annots** 126 0 R/**Contents** 123 0 R/MediaBox[ 0 0 595 842]/Parent 6 0 R/Resources 125 0 R>>  endobj |
| **155** 0 obj<</Type/Page/**Annots** 159 0 R/**Contents** 156 0 R/MediaBox[ 0 0 595 842]/Parent 6 0 R/Resources 158 0 R>>  endobj |
| **1496** 0 obj<</Type/Page/**Contents** 1497 0 R/MediaBox[ 0.000000 0.000000 595.000000 842.000000]/Parent 3 0 R/Resources<</ProcSet[/PDF/Text/ImageB/ImageC/ImageI]/XObject<</XOb1495 1495 0 R>>>>>>  endobj |
| **1499** 0 obj<</Type/Page/**Contents** 1500 0 R/MediaBox[ 0.000000 0.000000 595.000000 842.000000]/Parent 3 0 R/Resources<</ProcSet[/PDF/Text/ImageB/ImageC/ImageI]/XObject<</XOb1498 1498 0 R>>>>>>  endobj |
| …… |
| …… |
| …… |

由此可见，Pages里面的obj数组里面，既有Annots（**标注红色**），也有Contents（**标注蓝色**）。

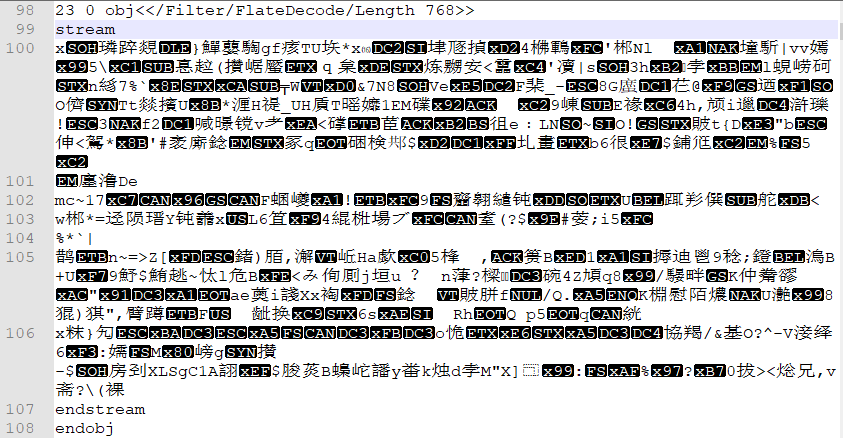
注意：如果是Annots，也会包含属性Contents（**标注绿色**）。

先看一个Annot定义（obj id：22）

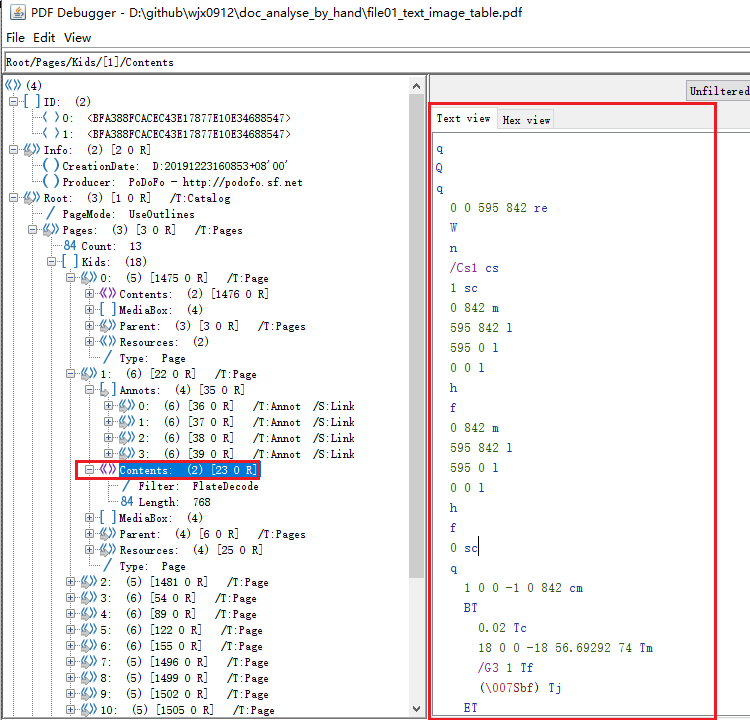
|  |
| --- |
| **22** 0 obj  <<  /Type /Page  /**Annots** 35 0 R  /**Contents** 23 0 R  /MediaBox[ 0 0 595 842]  /Parent 6 0 R  /Resources 25 0  R>>  endobj |
| 35 0 obj[ **36** 0 R 37 0 R 38 0 R 39 0 R]  endobj |
| **36** 0 obj  <<  /Type /Annot  /A **1386** 0 R  /Border[ 0 0 0]  /Rect[ 190.026100 665.500000 394.850800 670.500000]  /Subtype  /Link  >>  endobj |
| 37 0 obj<</Type/Annot/A 1384 0 R/Border[ 0 0 0]/Rect[ 190.026100 670.500000 394.850800 685]/Subtype/Link>>  endobj |
| 38 0 obj<</Type/Annot/A 1383 0 R/Border[ 0 0 0]/Rect[ 116.586100 646 246.408100 651]/Subtype/Link>>  endobj |
| 39 0 obj<</Type/Annot/A 1381 0 R/Border[ 0 0 0]/Rect[ 116.586100 651 246.408100 665.500000]/Subtype/Link>>  endobj |
| **1386** 0 obj<</Type/Action/S/URI/URI **1385** 0 R>>  endobj |
| **1385** 0 obj(https://github.com/qyuhen/book)  endobj |

这个Annot是一个超链接：https://github.com/qyuhen/book，实际在page2里面使用，后面再深入分析。

Annot-22里面也定义了Contents，它的id是23：



用pdfbox-debugger解码后内容如下：



有些复杂，放到后面分析。

Annot-22的parent的对象id是6（上面的Parent字段可以看到）

|  |
| --- |
| 6 0 obj  <<  /Type /Pages  /Count 8  /Kids [ 5 0 R 22 0 R 44 0 R 54 0 R 89 0 R 122 0 R 155 0 R 169 0 R]  /Parent **1362** 0 R  >>  endobj |
| **1362** 0 obj  <<  /Type /Pages  /Count 64  /Kids [ 6 0 R 176 0 R 219 0 R 262 0 R 301 0 R 340 0 R 375 0 R 414 0 R]  /Parent **1367** 0 R  >>  endobj |
| **1367** 0 obj  <<  /Type /Pages  /Count 265  /Kids [ 1362 0 R 1363 0 R 1364 0 R 1365 0 R 1366 0 R]  /MediaBox [ 0 0 595 842]  >>  endobj |

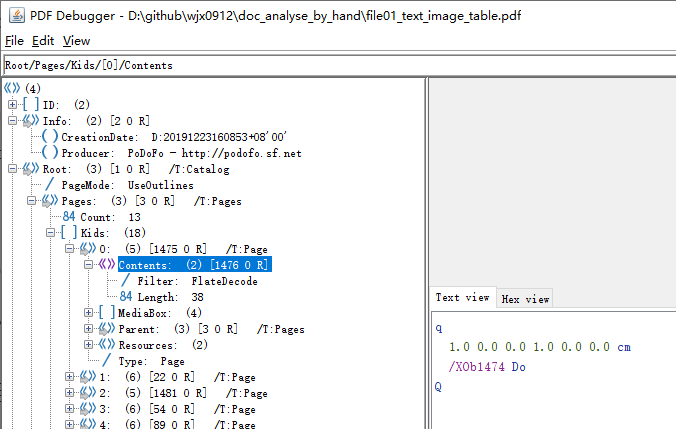
接下来看看第1个page的内容（obj id：1475），这个是重点

|  |
| --- |
| 1475 0 obj  <<  /Type /Page  /Contents 1476 0 R  /MediaBox[ 0.000000 0.000000 595.000000 842.000000]  /Parent 3 0 R  /Resources  <<  /ProcSet [/PDF /Text /ImageB /ImageC /ImageI]  /XObject <</**XOb1474** **1474** 0 R>>  >>  >>  endobj |
| 1476 0 obj<</Filter/FlateDecode/Length 38>>  stream  x???@  D垚?9桲??尚能D??? 髜l  endstream  endobj |

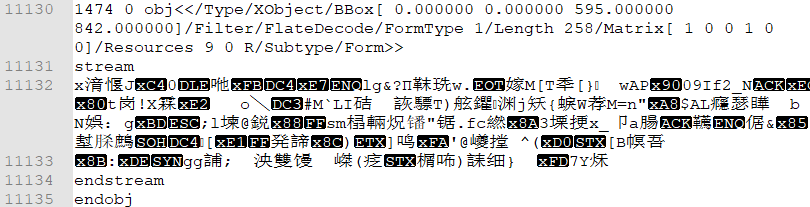
关于FlateDecode算法，ISO-32000里面描述如下：



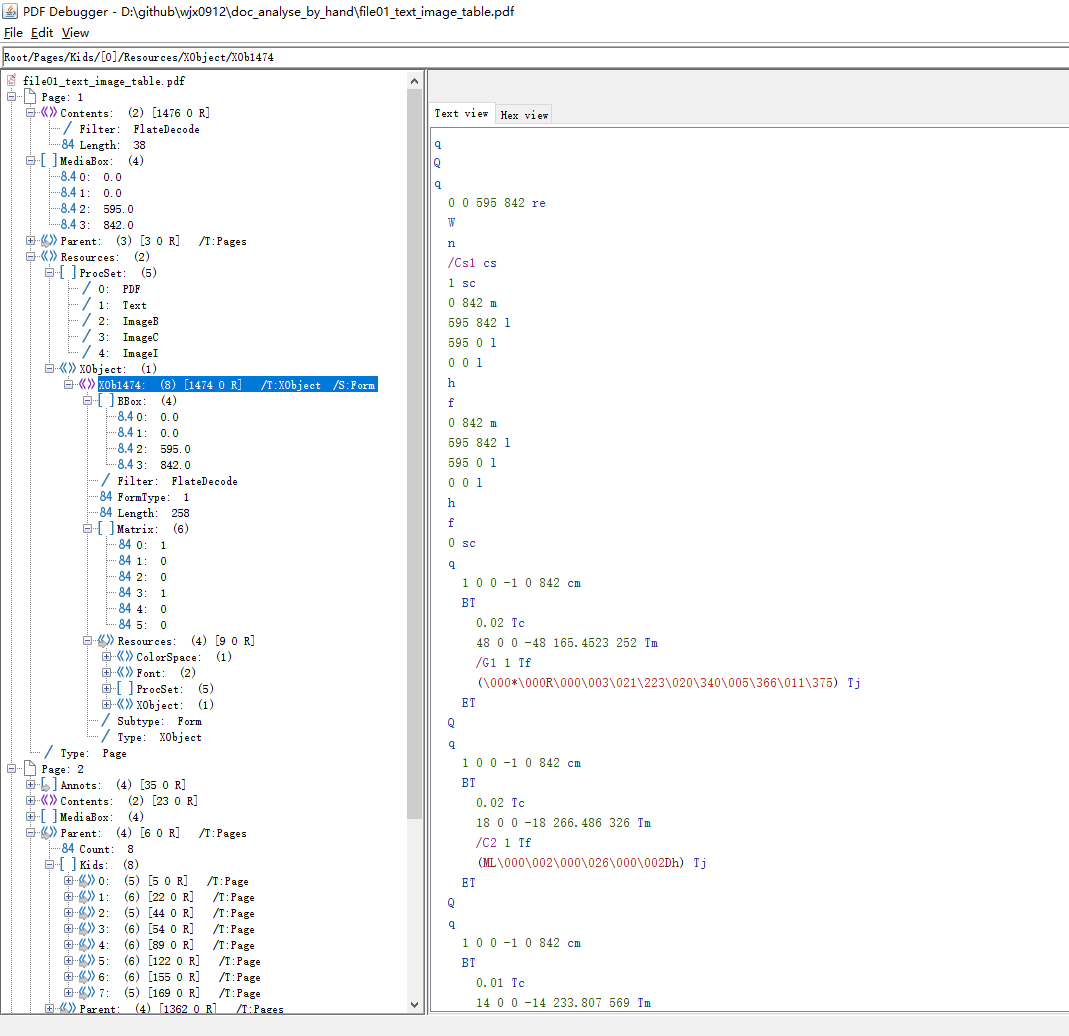
用pdfbox debugger解析1476：



接下来看对象1474：



pdfbox debugger解码如下：



里面的内容就是page1的内容，解码后的文字如下：

|  |
| --- |
| q  Q  q  0 0 595 842 re  W  n  /Cs1 cs  1 sc  0 842 m  595 842 l  595 0 l  0 0 l  h  f  0 842 m  595 842 l  595 0 l  0 0 l  h  f  0 sc  q  1 0 0 -1 0 842 cm  BT  0.02 Tc  48 0 0 -48 165.4523 252 Tm  /G1 1 Tf  **(\000\*\000R\000\003\021\223\020\340\005\366\011\375)** Tj  ET  Q  q  1 0 0 -1 0 842 cm  BT  0.02 Tc  18 0 0 -18 266.486 326 Tm  /C2 1 Tf  **(ML\000\002\000\026\000\002Dh)** Tj  ET  Q  q  1 0 0 -1 0 842 cm  BT  0.01 Tc  14 0 0 -14 233.807 569 Tm  /C2 1 Tf  **(+\235+\235-\206 \200\000\002\000\002\000\002+I+I&1 \*)** Tj  ET  Q  /Perceptual ri  q  35 0 0 68 280 168 cm  /Im1 Do  Q  Q |

BT 是begin text, ET是end text；里面的子属性里面有Tc, Tm, Tf, Tj，含义参考PDF- 32000 Table A.1

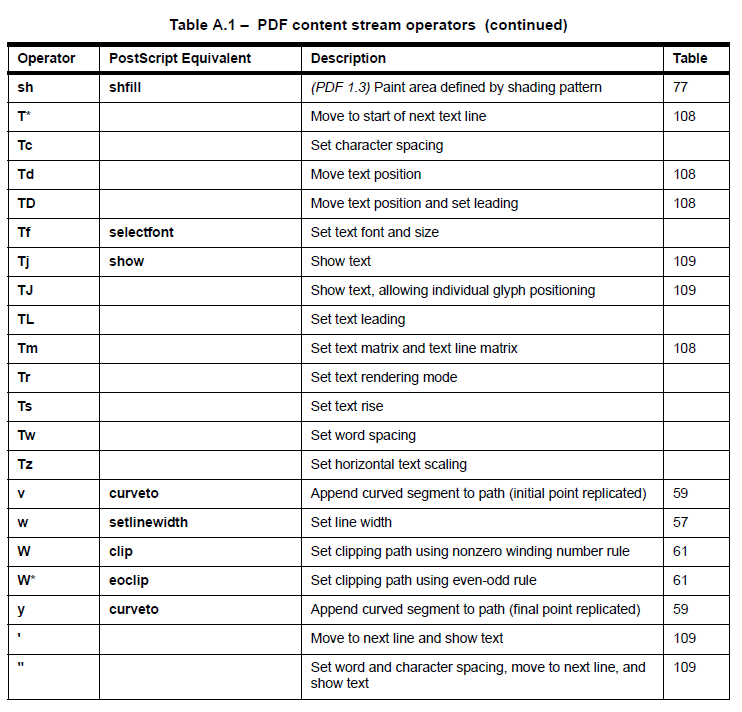
红色的3行是实体文字，分别是：

第1行：Go学习笔记

第2行：第4版

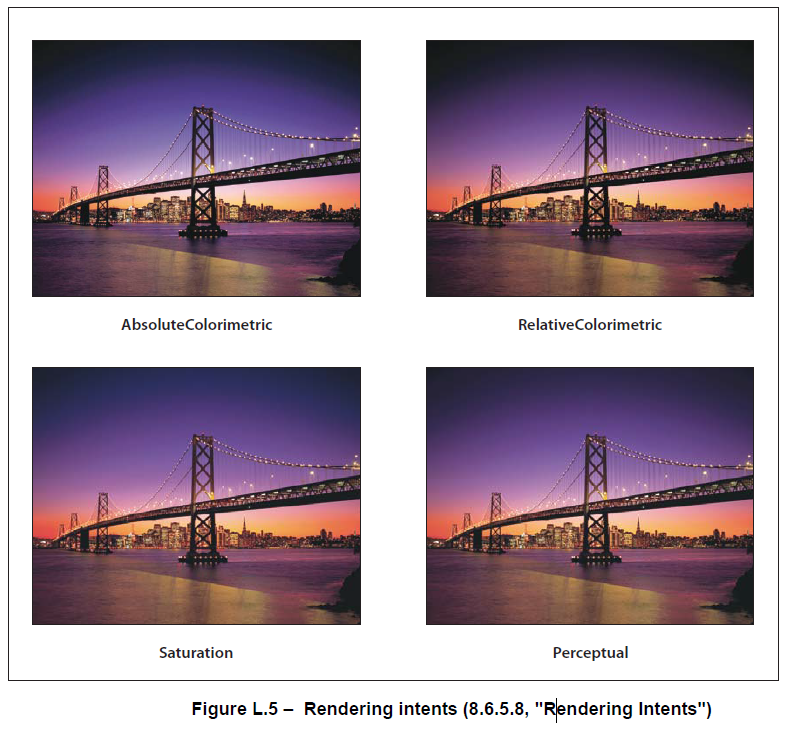
第3行：好好学习 天天向上

这三行红色文字应该是个引用，后面再继续分析。



Tj是需要显示的文字。

/Preceptual是配色方式，ISO 32000里面描述：



倒数第三行：/Im1 Do

Do是 (paint external object) ，这是一张图片。