**JPEG文件储存格式**

本文摘取自<http://www.ibm.com/developerworks/cn/linux/l-cn-jpeg/> JPEG 原理详细实例分析及其在嵌入式 Linux 中的应用

关于JPEG的其他内容请参考原文。

四．JPEG文件存储格式

介绍了 JPEG 的原理，我们再来结合一个具体的实例来详细讨论上面所涉及到的细节。

我们先来制作一个简单的8X8大小的像素图，然后把它存成JPEG格式。

方法是用 windows 的画图工具，定义一个8X8大小的图，用一些色块填充进去，然后另存为 JPEG 格式，如 test8x8.jpg。如下图所示：

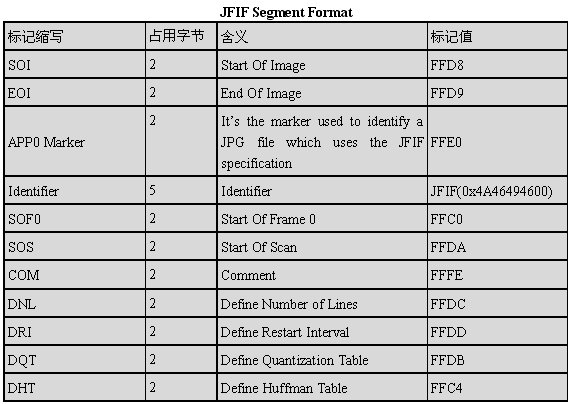
 

保存成的文件后缀为 jpg，但按标准来说，它是一种 JFIF 格式标准的文件，里面的图像的压缩方式是 JPEG。

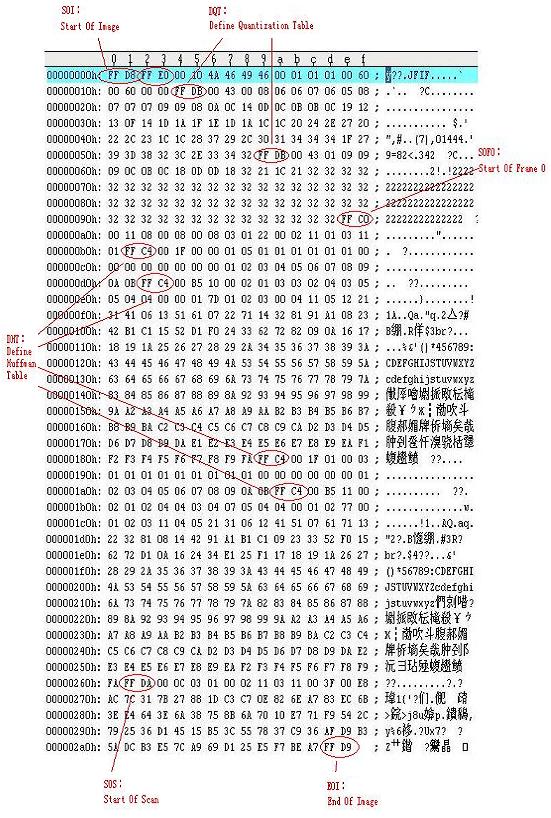
JFIF 是一个文件格式标准，JPEG 是一个压缩标准，总体来说它们不是一个概念。

JFIF 是 JPEG File Interchange Format 的缩写，也即 JPEG 文件交换格式。JFIF 是一个图片文件格式标准，它是一种使用 JPEG 图像压缩技术存储摄影图像的方法。JFIF 代表了一种"通用语言"文件格式，它是专门为方便用户在不同的计算机和应用程序间传输 JPEG 图像而设计的语言。

JFIF 文件格式定义了一些内容是 JPEG 压缩标准未定义的，如 resolution/aspect ratio，color space 等。

我们可以打开 JPEG 文件查看里面的内容，即可看到上面的各个标记段：

从图上可以看出：

在头部有 FFD8 ，表示图像的开始；结束部分有 FFD9 ，表示图像的结束。

在中间有两个量化表 DQT 对应的标记 FFDB ；

还有图像大小信息对应的 FFC0

再后面有四个 Haffman 表对应的 FFC4 ；

一般一个 JPG 文件里会有 2 类 Haffman 表：一个用于 DC 一个用于 AC ，也即实际有 4个表，亮度的 DC，AC 两个，色度的 DC，AC 两个。

然后是图像数据段标记 FFDA；

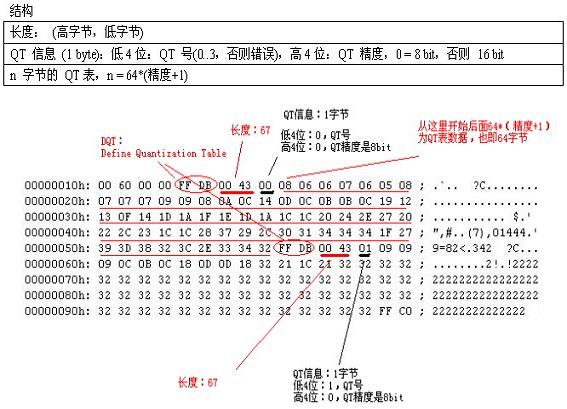
我们再来看看各个标记的细部，具体分析一下各个部分的含义。

**1、图片的识别信息**

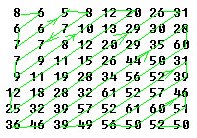
上面的内容，在标记 FFE0 后，即为长度16。然后是5字节的 JFIF 标识符号，说明这是一个 JPEG 压缩的文件。然后是主/次版本号码。下一个为 XY 像素的单位，这里为1，表示单位为点数/英寸。然后是 XY 方向的像素密度，这里是 96DPI，最后是缩略图有关信息，这里为0。

**2、量化表的实例**

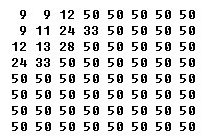
上面这个内容，FFDB 标记后的长度值为67，接下来的是 QT 信息，占一个字节；这里是0，表示这个 QT 表编号为0，并且精度是8bit。然后后面就是64个8x8的 QT 表的各个 item 了。

也即第一个 DQT 量化表的内容表示为十进制是：

这个表即为 JPEG 亮度量化表。

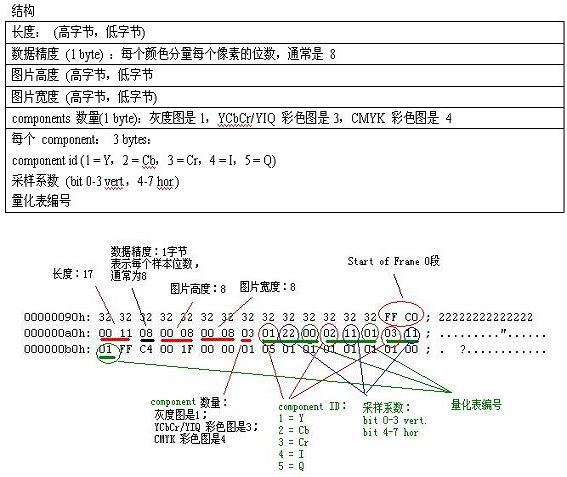
第二个量化表的内容为：

这个表的内容即为 JPEG 色度量化表。

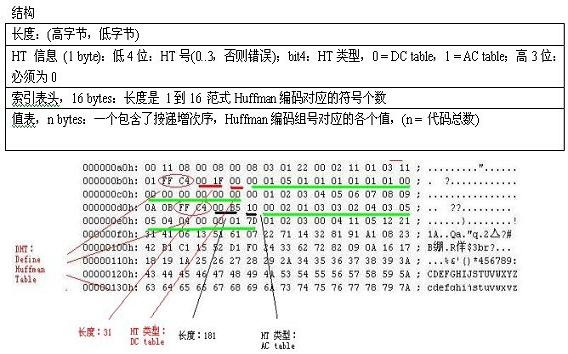
当你打开不同的 JPEG 文件，你会看到这两个表可能也是会有区别的。这个主要是使用了不同的量化方式的结果。

**3、图像信息段**

上面这个内容，FFC0 标记后即是长度，为17，然后是一个字节的数据精度，通常是为8，代表样本位数。接下来是图片的高度，占两字节，这里即为8，然后是图片的宽度，也为8，这也就是我们定义的8x8的内容。然后是 component 的个数，这里是3，表示 YUV。接下来是三组数据，每组数据里，第一个是 component ID，第二个是采样系数，这里 Y 的采样系数为22，说明垂直是2，水平是2。再后面就是量化表的编号了。

**4、Haffman 表的实例**

上面这个内容，FFC4 标记后的内容为数据长度，再接着的1字节为 Huffman Table 的信息，低4位是 HT ID 号，第5位是 HT 表类型标记，再高三位是为0。

第一个 DHT 表，00，类型为 DC table，HT ID 号为 0；

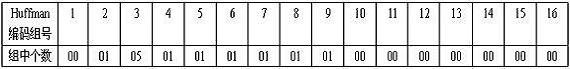
第二个 DHT 表，10，类型为 AC table，HT ID 号也为 0；

第三个 DHT 表，01，类型为 DC table，HT ID 号为 1；

第四个 DHT 表，11，类型为 AC table，HT ID 号为 1；

即前两个表为Y亮度分量的 DC/AC 表，后两个为 UV 色度分量的 DC/AC 表。

以第一个表为例，因为长度只有 31，那么 00 后面的 16 字节，即绿色部分：

组号为 1 的组中，代码有 0 个；

组号为 2 的，代码有 1 个；

组号为 3 的代码有 5 个；

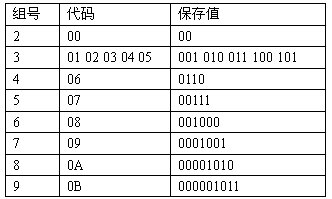
组号为 4/5/6/7/8/9 的代码各 1 个。

总共 12 个。

再看后续的数据：

00 01 02 03 04 05 06 07 08 09 0A 0B

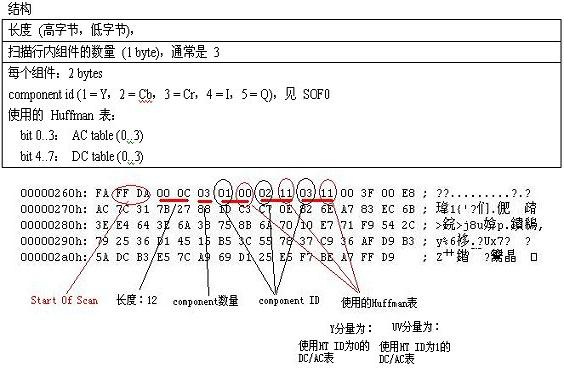
即对应：

其他未出现的组号，对应的数据未使用到。也就是说前面提到过的范式 Huffman 编码里，目前只使用部分数据即可，原因是这个 8x8 的图像数据很小。

第二个 DHT 表就更复杂些了，长度有 181。

**5、图像数据段**

这里 SOS 段，长度为 12，后面所含有的 component 数量为 3 个，也即 Y UV。然后后面是各 component 的编号，及对应所使用的 Huffman 表的 ID 是多少。

在这个段的后面就是所有压缩后的数据。直到结束的问题，即 FFD9，EOI（End Of Image）。