

MediaInfo源代码分析 5：JPEG解析代码分析

2013年10月14日 16:39:56 阅读数：4270

MediaInfo源代码分析系列文章列表：

[MediaInfo源代码分析 1：整体结构](#)

[MediaInfo源代码分析 2：API函数](#)

[MediaInfo源代码分析 3：Open\(\)函数](#)

[MediaInfo源代码分析 4：Inform\(\)函数](#)

[MediaInfo源代码分析 5：JPEG解析代码分析](#)

本文分析MediaInfo中解码JPEG信息的模块。之前写了几篇文章都是关于MediaInfo主程序的，并没有分析其具体是如何解析不同多媒体文件信息的。在这里分析一下解码JPEG文件的代码。其他格式如BMP，GIF等解析的思路基本上是类似的。

File_Jpeg.h的File_Jpeg类的定义如下所示：

```
[cpp]
1. //*****
2. // Class File_Jpeg
3. //*****
4. //继承 File_Analyze
5. class File_Jpeg : public File_Analyze
6. {
7. public :
8.     //In
9.     stream_t StreamKind;
10.    bool    Interlaced;
11.
12.    //Constructor/Destructor
13.    File_Jpeg();
14.
15. private :
16.     //Streams management
17.     void Streams_Accept();
18.
19.     //Buffer - File header
20.     bool FileHeader_Begin();
21.
22.     //Buffer - Synchro
23.     bool Synchronize();
24.     bool Synched_Test();
25.     void Synched_Init();
26.
27.     //Buffer - Demux
28.     #if MEDIAINFO_DEMUX
29.     bool Demux_UnpacketizeContainer_Test() {return Demux_UnpacketizeContainer_Test_OneFramePerFile();}
30.     #endif //MEDIAINFO_DEMUX
31.
32.     //Buffer - Global
33.     void Read_Buffer_Unsynched();
34.     #if MEDIAINFO_SEEK
35.     size_t Read_Buffer_Seek (size_t Method, int64u Value, int64u ID) {return Read_Buffer_Seek_OneFramePerFile(Method, Value, ID);}
36.     #endif //MEDIAINFO_SEEK
37.
38.     //Buffer - Per element
39.     //解析头
40.     void Header_Parse();
41.     bool Header_Parser_Fill_Size();
42.     //解析数据
43.     void Data_Parse();
44.
45.     //Elements
46.     //JPEG中的单元
47.     //解析相应的单元，并获得信息
48.     void TEM () {};
49.     void SOC () {}
50.     void SIZ ();
51.     void COD ();
52.     void COC () {Skip_XX(Element_Size, "Data");}
53.     void TLM () {Skip_XX(Element_Size, "Data");}
54.     void PLM () {Skip_XX(Element_Size, "Data");}
55.     void PLT () {Skip_XX(Element_Size, "Data");}
56.     void QCD ();
57.     void QCC () {Skip_XX(Element_Size, "Data");}
58.     void RGN () {Skip_XX(Element_Size, "Data");}
59.     void PPM () {Skip_XX(Element_Size, "Data");}
```

```

60. void PPT () {Skip_XX(Element_Size, "Data");}
61. void CME () {Skip_XX(Element_Size, "Data");}
62. void SOT () {Skip_XX(Element_Size, "Data");}
63. void SOP () {Skip_XX(Element_Size, "Data");}
64. void EPH () {Skip_XX(Element_Size, "Data");}
65. void SOD ();
66. void SOF ();
67. void SOF0() {SOF_()};
68. void SOF1() {SOF_()};
69. void SOF2() {SOF_()};
70. void SOF3() {SOF_()};
71. void DHT () {Skip_XX(Element_Size, "Data");}
72. void SOF5() {SOF_()};
73. void SOF6() {SOF_()};
74. void SOF7() {SOF_()};
75. void JPG () {Skip_XX(Element_Size, "Data");}
76. void SOF9() {SOF_()};
77. void SOFA() {SOF_()};
78. void SOFB() {SOF_()};
79. void DAC () {Skip_XX(Element_Size, "Data");}
80. void SOFD() {SOF_()};
81. void SOFE() {SOF_()};
82. void SOFF() {SOF_()};
83. void RST0() {};
84. void RST1() {};
85. void RST2() {};
86. void RST3() {};
87. void RST4() {};
88. void RST5() {};
89. void RST6() {};
90. void RST7() {};
91. void SOI () {};
92. void EOI () {};
93. void SOS ();
94. void DQT () {Skip_XX(Element_Size, "Data");}
95. void DNL () {Skip_XX(Element_Size, "Data");}
96. void DRI () {Skip_XX(Element_Size, "Data");}
97. void DHP () {Skip_XX(Element_Size, "Data");}
98. void EXP () {Skip_XX(Element_Size, "Data");}
99. void APP0();
100. void APP0_AVI1();
101. void APP0_JFIF();
102. void APP0_JFFF();
103. void APP0_JFFF_JPEG();
104. void APP0_JFFF_1B();
105. void APP0_JFFF_3B();
106. void APP1();
107. void APP1_EXIF();
108. void APP2() {Skip_XX(Element_Size, "Data");}
109. void APP3() {Skip_XX(Element_Size, "Data");}
110. void APP4() {Skip_XX(Element_Size, "Data");}
111. void APP5() {Skip_XX(Element_Size, "Data");}
112. void APP6() {Skip_XX(Element_Size, "Data");}
113. void APP7() {Skip_XX(Element_Size, "Data");}
114. void APP8() {Skip_XX(Element_Size, "Data");}
115. void APP9() {Skip_XX(Element_Size, "Data");}
116. void APPA() {Skip_XX(Element_Size, "Data");}
117. void APPB() {Skip_XX(Element_Size, "Data");}
118. void APPC() {Skip_XX(Element_Size, "Data");}
119. void APPD() {Skip_XX(Element_Size, "Data");}
120. void APPE();
121. void APPE_Adobe0();
122. void APPF() {Skip_XX(Element_Size, "Data");}
123. void JPG0() {Skip_XX(Element_Size, "Data");}
124. void JPG1() {Skip_XX(Element_Size, "Data");}
125. void JPG2() {Skip_XX(Element_Size, "Data");}
126. void JPG3() {Skip_XX(Element_Size, "Data");}
127. void JPG4() {Skip_XX(Element_Size, "Data");}
128. void JPG5() {Skip_XX(Element_Size, "Data");}
129. void JPG6() {Skip_XX(Element_Size, "Data");}
130. void JPG7() {Skip_XX(Element_Size, "Data");}
131. void JPG8() {Skip_XX(Element_Size, "Data");}
132. void JPG9() {Skip_XX(Element_Size, "Data");}
133. void JPGA() {Skip_XX(Element_Size, "Data");}
134. void JRGB() {Skip_XX(Element_Size, "Data");}
135. void JGRC() {Skip_XX(Element_Size, "Data");}
136. void JGRC() {Skip_XX(Element_Size, "Data");}
137. void JGRC() {Skip_XX(Element_Size, "Data");}
138. void COM () {Skip_XX(Element_Size, "Data");}
139. //Temp
140. int8u APPE_Adobe0_transform;
141. bool APP0_JFIF_Parsed;
142. bool SOS_SOD_Parsed;
143. };

```

上面代码有以下几个特点：

1.继承了File__Analyze类

2.包含了很多JPEG中的数据单元的解析：DHT(), DQT()等等

下面来分别仔细看看源代码：

1.File__Analyze类代码巨多无比，先不分析。他继承了File__Base

2.看一个解码具体单元的代码：SOF_()

注：SOF0（Start of Image，图像开始）。

SOF0，Start of Frame，帧图像开始

标记代码 2字节 固定值0xFFC0

包含9个具体字段：

① 数据长度 2字节 ①~⑥六个字段的总长度

即不包括标记代码，但包括本字段

② 精度 1字节 每个数据样本的位数

通常是8位，一般软件都不支持 12位和16位

③ 图像高度 2字节 图像高度（单位：像素），如果不支持 DNL 就必须 >0

④ 图像宽度 2字节 图像宽度（单位：像素），如果不支持 DNL 就必须 >0

⑤ 颜色分量数 1字节 只有3个数值可选

1：灰度图；3：YCrCb或YIQ；4：CMYK

而JFIF中使用YCrCb，故这里颜色分量数恒为3

⑥颜色分量信息 颜色分量数×3字节（通常为9字节）

a) 颜色分量ID 1字节

b) 水平/垂直采样因子 1字节 高4位：水平采样因子

低4位：垂直采样因子

（曾经看到某资料把这两者调转了）

c) 量化表 1字节 当前分量使用的量化表的ID

本标记段中，字段⑥应该重复出现，有多少个颜色分量（字段⑤），就出现多少次（一般为3次）。

=====

```
[cpp]
1. void File_Jpeg::SOF_()
2. {
3.     //Parsing
4.     vector<Jpeg_samplingfactor> SamplingFactors;
5.     int16u Height, Width;
6.     int8u Resolution, Count;
7.     Get_B1 (Resolution, "P - Sample precision");
8.     Get_B2 (Height, "Y - Number of lines");
9.     Get_B2 (Width, "X - Number of samples per line");
10.    Get_B1 (Count, "Nf - Number of image components in frame");
11.    for (int8u Pos=0; Pos<Count; Pos++)
12.    {
13.        Jpeg_samplingfactor SamplingFactor;
14.        Element_Begin1("Component");
15.        Get_B1 ( SamplingFactor.Ci, "Ci - Component identifier"); if (SamplingFactor.Ci>Count) Element_In
16.        fol(Ztring().append(1, (Char)SamplingFactor.Ci)); else Element_Info1(SamplingFactor.Ci);
17.        BS_Begin();
18.        Get_S1 (4, SamplingFactor.Hi, "Hi - Horizontal sampling factor"); Element_Info1(SamplingFactor.Hi);
19.        Get_S1 (4, SamplingFactor.Vi, "Vi - Vertical sampling factor"); Element_Info1(SamplingFactor.Vi);
20.        BS_End();
21.        Skip_B1( "Tqi - Quantization table destination selector");
22.        Element_End0();
23.        //Filling list of HiVi
24.        SamplingFactors.push_back(SamplingFactor);
25.    }
26.
27.    FILLING_BEGIN_PRECISE();
28.    if (Frame_Count==0 && Field_Count==0)
29.    {
30.        Accept("JPEG");
31.        Fill("JPEG");
32.
33.        if (Count_Get(StreamKind_Last)==0)
34.            Stream_Prepare(StreamKind_Last);
35.        Fill(StreamKind_Last, 0, Fill_Parameter(StreamKind_Last, Generic_Format), "JPEG");
36.        Fill(StreamKind_Last, 0, Fill_Parameter(StreamKind_Last, Generic_Codec), "JPEG");
    }
```

```

37.         if (StreamKind_Last==Stream_Image)
38.             Fill(Stream_Image, 0, Image_Codec_String, "JPEG", Unlimited, true, true); //To Avoid automatic filling
39.         if (StreamKind_Last==Stream_Video)
40.             Fill(Stream_Video, 0, Video_InternetMediaType, "video/JPEG", Unlimited, true, true);
41.         Fill(StreamKind_Last, 0, Fill_Parameter(StreamKind_Last, Generic_BitDepth), Resolution);
42.         Fill(StreamKind_Last, 0, "Height", Height*(Interlaced?2:1));
43.         Fill(StreamKind_Last, 0, "Width", Width);
44.
45.         //ColorSpace from http://docs.oracle.com/javase/1.4.2/docs/api/javax/imageio/metadata/doc-files/jpeg_metadata.html
46.         switch (APPE_Adobe0_transform)
47.         {
48.             case 0x01 :
49.                 if (Count==3)
50.                     Fill(StreamKind_Last, 0, "ColorSpace", "YUV");
51.             case 0x02 :
52.                 if (Count==4)
53.                     Fill(StreamKind_Last, 0, "ColorSpace", "YCCB");
54.                 break;
55.             default :
56.                 {
57.                     int8u Ci[256];
58.                     memset(Ci, 0, 256);;
59.                     for (int8u Pos=0; Pos<Count; Pos++)
60.                         Ci[SamplingFactors[Pos].Ci]++;
61.
62.                     switch (Count)
63.                     {
64.                         case 1 : Fill(StreamKind_Last, 0, "ColorSpace", "Y"); break;
65.                         case 2 : Fill(StreamKind_Last, 0, "ColorSpace", "YA"); break;
66.                         case 3 :
67.                             if (!APP0_JFIF_Parsed && Ci['R']==1 && Ci['G']==1 && Ci['B']==1)
68.                                 //RGB
69.                                 Fill(StreamKind_Last, 0, "ColorSpace", "RGB");
70.                             else if ((Ci['Y']==1 && ((Ci['C']==1 && Ci['c']==1)
71.                                 //YCC
72.                                 || Ci['C']==2))
73.                                 //YCC
74.                                 || APP0_JFIF_Parsed
75.                                 //APP0 JFIF header present so YCC
76.                                 || APPE_Adobe0_transform==0
77.                                 //transform set to YCC
78.                                 || (SamplingFactors[0].Ci==0 && SamplingFactors[1].Ci==1 && SamplingFactors[2].Ci==
79.                                 //012
80.                                 || (SamplingFactors[0].Ci==1 && SamplingFactors[1].Ci==2 && SamplingFactors[2].Ci==
81.                                 //123
82.                                 Fill(StreamKind_Last, 0, "ColorSpace", "YUV");
83.                             break;
84.                         case 4 :
85.                             if (!APP0_JFIF_Parsed && Ci['R']==1 && Ci['G']==1 && Ci['B']==1 && Ci['A']==1)
86.                                 //RGBA
87.                                 Fill(StreamKind_Last, 0, "ColorSpace", "RGBA");
88.                             else if ((Ci['Y']==1 && Ci['A']==1 && ((Ci['C']==1 && Ci['c']==1)
89.                                 //YCCA
90.                                 || Ci['C']==2))
91.                                 //YCCA
92.                                 || APP0_JFIF_Parsed
93.                                 //APP0 JFIF header present so YCCA
94.                                 || (SamplingFactors[0].Ci==0 && SamplingFactors[1].Ci==1 && SamplingFactors[2].Ci==
95.                                 & SamplingFactors[3].Ci==3) //0123
96.                                 || (SamplingFactors[0].Ci==1 && SamplingFactors[1].Ci==2 && SamplingFactors[2].Ci==
97.                                 & SamplingFactors[3].Ci==4)) //1234
98.                                 Fill(StreamKind_Last, 0, "ColorSpace", "YUVA");
99.                             else if (APPE_Adobe0_transform==0)
100.                                 //transform set to CMYK
101.                                 Fill(StreamKind_Last, 0, "ColorSpace", "YCCB");
102.                             break;
103.                         default: ;
104.                     }
105.                 }
106.             }
107.
108.         //Chroma subsampling
109.         if ((SamplingFactors.size()==3 || SamplingFactors.size()==4) && SamplingFactors[1].Hi==1 && SamplingFactors[2].Hi==1 && S
110.         lingFactors[1].Vi==1 && SamplingFactors[2].Vi==1)
111.         {
112.             string ChromaSubsampling;
113.             switch (SamplingFactors[0].Hi)
114.             {
115.                 case 1 :
116.                     switch (SamplingFactors[0].Vi)
117.                     {
118.                         case 1 : ChromaSubsampling="4:4:4"; break;
119.                         default: ;
120.                     }
121.                     break;
122.                 case 2 :
123.                     switch (SamplingFactors[0].Vi)
124.                     {
125.                         case 1 : ChromaSubsampling="4:2:2"; break;
126.                         case 2 : ChromaSubsampling="4:2:0"; break;
127.                         default: ;

```

```

113.         }
114.         break;
115.     case 4 :
116.         switch (SamplingFactors[0].Vi)
117.         {
118.             case 1 : ChromaSubsampling="4:1:1"; break;
119.             default: ;
120.         }
121.         break;
122.     default: ;
123. }
124. if (!ChromaSubsampling.empty())
125. {
126.     if (SamplingFactors.size()==4)
127.     {
128.         if (ChromaSubsampling=="4:4:4" && SamplingFactors[3].Hi==1 && SamplingFactors[3].Vi==1)
129.             ChromaSubsampling+=":4";
130.         else
131.             ChromaSubsampling+=":?" ;
132.     }
133.     Fill(StreamKind_Last, 0, "ChromaSubsampling", ChromaSubsampling);
134. }
135. }
136. }
137. FILLING_END();
138. }

```

从代码的含义可知，提取出了图像的宽，高，采样方式等信息。
详细的代码暂时没有时间研究了，先这样了。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/12452991>

文章标签： [MediaInfo](#) [源代码](#) [JPEG](#) [解析](#)

个人分类： [MediaInfo](#)

所属专栏： [开源多媒体项目源代码分析](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com