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：4264

==========

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
//***************************************************************************
// Class File_Jpeg
//***************************************************************************
//继承 File__Analyze
class File_Jpeg : public File__Analyze
{
public :
    //In
    stream_t StreamKind;
    bool     Interlaced;

    //Constructor/Destructor
    File_Jpeg();

private :
    //Streams management
    void Streams_Accept();

    //Buffer - File header
    bool FileHeader_Begin();

    //Buffer - Synchro
    bool Synchronize();
    bool Synched_Test();
    void Synched_Init();

    //Buffer - Demux
    #if MEDIAINFO_DEMUX
    bool Demux_UnpacketizeContainer_Test() {return Demux_UnpacketizeContainer_Test_OneFramePerFile();}
    #endif //MEDIAINFO_DEMUX

    //Buffer - Global
    void Read_Buffer_Unsynched();
    #if MEDIAINFO_SEEK
    size_t Read_Buffer_Seek (size_t Method, int64u Value, int64u ID) {return Read_Buffer_Seek_OneFramePerFile(Method, Value, ID);}
    #endif //MEDIAINFO_SEEK

    //Buffer - Per element
    //解析头
    void Header_Parse();
    bool Header_Parser_Fill_Size();
    //解析数据
    void Data_Parse();

    //Elements
    //JPEG中的单元
    //解析相应的单元，并获得信息
    void TEM () {};
    void SOC () {}
    void SIZ ();
    void COD ();
    void COC () {Skip_XX(Element_Size, "Data");}
    void TLM () {Skip_XX(Element_Size, "Data");}
    void PLM () {Skip_XX(Element_Size, "Data");}
    void PLT () {Skip_XX(Element_Size, "Data");}
    void QCD ();
    void QCC () {Skip_XX(Element_Size, "Data");}
    void RGN () {Skip_XX(Element_Size, "Data");}
    void PPM () {Skip_XX(Element_Size, "Data");}
```

```
    void PPT () {Skip_XX(Element_Size, "Data");}
    void CME () {Skip_XX(Element_Size, "Data");}
    void SOT () {Skip_XX(Element_Size, "Data");}
    void SOP () {Skip_XX(Element_Size, "Data");}
    void EPH () {Skip_XX(Element_Size, "Data");}
    void SOD ();
    void SOF_();
    void S0F0() {SOF_();};
    void S0F1() {SOF_();};
    void S0F2() {SOF_();};
    void S0F3() {SOF_();}
    void DHT () {Skip_XX(Element_Size, "Data");}
    void S0F5() {SOF_();}
    void S0F6() {SOF_();}
    void S0F7() {SOF_();}
    void JPG () {Skip_XX(Element_Size, "Data");}
    void S0F9() {SOF_();}
    void S0FA() {SOF_();}
    void S0FB() {SOF_();}
    void DAC () {Skip_XX(Element_Size, "Data");}
    void S0FD() {SOF_();}
    void S0FE() {SOF_();}
    void S0FF() {SOF_();}
    void RST0() {};
    void RST1() {};
    void RST2() {};
    void RST3() {};
    void RST4() {};
    void RST5() {};
    void RST6() {};
    void RST7() {};
    void SOI () {};
    void EOI () {};
    void SOS ();
    void DQT () {Skip_XX(Element_Size, "Data");}
    void DNL () {Skip_XX(Element_Size, "Data");}
    void DRI () {Skip_XX(Element_Size, "Data");}
    void DHP () {Skip_XX(Element_Size, "Data");}
    void EXP () {Skip_XX(Element_Size, "Data");}
    void APP0();
    void APP0_AVI1();
    void APP0_JFIF();
    void APP0_JFFF();
    void APP0_JFFF_JPEG();
    void APP0_JFFF_1B();
    void APP0_JFFF_3B();
    void APP1();
    void APP1_EXIF();
    void APP2() {Skip_XX(Element_Size, "Data");}
    void APP3() {Skip_XX(Element_Size, "Data");}
    void APP4() {Skip_XX(Element_Size, "Data");}
    void APP5() {Skip_XX(Element_Size, "Data");}
    void APP6() {Skip_XX(Element_Size, "Data");}
    void APP7() {Skip_XX(Element_Size, "Data");}
    void APP8() {Skip_XX(Element_Size, "Data");}
    void APP9() {Skip_XX(Element_Size, "Data");}
    void APPA() {Skip_XX(Element_Size, "Data");}
    void APPB() {Skip_XX(Element_Size, "Data");}
    void APPC() {Skip_XX(Element_Size, "Data");}
    void APPD() {Skip_XX(Element_Size, "Data");}
    void APPE();
    void APPE_Adobe0();
    void APPF() {Skip_XX(Element_Size, "Data");}
    void JPG0() {Skip_XX(Element_Size, "Data");}
    void JPG1() {Skip_XX(Element_Size, "Data");}
    void JPG2() {Skip_XX(Element_Size, "Data");}
    void JPG3() {Skip_XX(Element_Size, "Data");}
    void JPG4() {Skip_XX(Element_Size, "Data");}
    void JPG5() {Skip_XX(Element_Size, "Data");}
    void JPG6() {Skip_XX(Element_Size, "Data");}
    void JPG7() {Skip_XX(Element_Size, "Data");}
    void JPG8() {Skip_XX(Element_Size, "Data");}
    void JPG9() {Skip_XX(Element_Size, "Data");}
    void JPGA() {Skip_XX(Element_Size, "Data");}
    void JPGB() {Skip_XX(Element_Size, "Data");}
    void JPGC() {Skip_XX(Element_Size, "Data");}
    void JPGD() {Skip_XX(Element_Size, "Data");}
    void COM () {Skip_XX(Element_Size, "Data");}

    //Temp
    int8u APPE_Adobe0_transform;
    bool  APP0_JFIF_Parsed;
    bool  SOS_SOD_Parsed;
};
```

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
void File_Jpeg::SOF_()
{
    //Parsing
    vector<Jpeg_samplingfactor> SamplingFactors;
    int16u Height, Width;
    int8u  Resolution, Count;
    Get_B1 (Resolution,                                         "P - Sample precision");
    Get_B2 (Height,                                             "Y - Number of lines");
    Get_B2 (Width,                                              "X - Number of samples per line");
    Get_B1 (Count,                                              "Nf - Number of image components in frame");
    for (int8u Pos=0; Pos<Count; Pos++)
    {
        Jpeg_samplingfactor SamplingFactor;
        Element_Begin1("Component");
        Get_B1 (   SamplingFactor.Ci,                           "Ci - Component identifier"); if (SamplingFactor.Ci>Count) Element_Info1(Ztring().append(1, (Char)SamplingFactor.Ci)); else Element_Info1(SamplingFactor.Ci);
        BS_Begin();
        Get_S1 (4, SamplingFactor.Hi,                           "Hi - Horizontal sampling factor"); Element_Info1(SamplingFactor.Hi);
        Get_S1 (4, SamplingFactor.Vi,                           "Vi - Vertical sampling factor"); Element_Info1(SamplingFactor.Vi);
        BS_End();
        Skip_B1(                                                "Tqi - Quantization table destination selector");
        Element_End0();

        //Filling list of HiVi
        SamplingFactors.push_back(SamplingFactor);
    }

    FILLING_BEGIN_PRECISE();
        if (Frame_Count==0 && Field_Count==0)
        {
            Accept("JPEG");
            Fill("JPEG");

            if (Count_Get(StreamKind_Last)==0)
                Stream_Prepare(StreamKind_Last);
            Fill(StreamKind_Last, 0, Fill_Parameter(StreamKind_Last, Generic_Format), "JPEG");
            Fill(StreamKind_Last, 0, Fill_Parameter(StreamKind_Last, Generic_Codec), "JPEG");
```

```
            if (StreamKind_Last==Stream_Image)
                Fill(Stream_Image, 0, Image_Codec_String, "JPEG", Unlimited, true, true); //To Avoid automatic filling
            if (StreamKind_Last==Stream_Video)
                Fill(Stream_Video, 0, Video_InternetMediaType, "video/JPEG", Unlimited, true, true);
            Fill(StreamKind_Last, 0, Fill_Parameter(StreamKind_Last, Generic_BitDepth), Resolution);
            Fill(StreamKind_Last, 0, "Height", Height*(Interlaced?2:1));
            Fill(StreamKind_Last, 0, "Width", Width);

            //ColorSpace from http://docs.oracle.com/javase/1.4.2/docs/api/javax/imageio/metadata/doc-files/jpeg_metadata.html
            switch (APPE_Adobe0_transform)
            {
                case 0x01 :
                            if (Count==3)
                                Fill(StreamKind_Last, 0, "ColorSpace", "YUV");
                case 0x02 :
                            if (Count==4)
                                Fill(StreamKind_Last, 0, "ColorSpace", "YCCB");
                            break;
                default   :
                            {
                            int8u Ci[256];
                            memset(Ci, 0, 256);;
                            for (int8u Pos=0; Pos<Count; Pos++)
                                Ci[SamplingFactors[Pos].Ci]++;

                            switch (Count)
                            {
                                case 1 :    Fill(StreamKind_Last, 0, "ColorSpace", "Y"); break;
                                case 2 :    Fill(StreamKind_Last, 0, "ColorSpace", "YA"); break;
                                case 3 :
                                                if (!APP0_JFIF_Parsed && Ci['R']==1 && Ci['G']==1 && Ci['B']==1)
                                    //RGB
                                                    Fill(StreamKind_Last, 0, "ColorSpace", "RGB");
                                                else if ((Ci['Y']==1 && ((Ci['C']==1 && Ci['c']==1)
                                    //YCc
                                                                                || Ci['C']==2))
                                                  //YCC
                                                    || APP0_JFIF_Parsed
                                                //APP0 JFIF header present so YCC
                                                    || APP0_transform==0
                                                //transform set to YCC
                                                    || (SamplingFactors[0].Ci==0 && SamplingFactors[1].Ci==1 && SamplingFactors[2].Ci==
                                                //012
                                                    || (SamplingFactors[0].Ci==1 && SamplingFactors[1].Ci==2 && SamplingFactors[2].Ci==
                                                //123
                                                    Fill(StreamKind_Last, 0, "ColorSpace", "YUV");
                                                break;
                                case 4 :
                                                 if (!APP0_JFIF_Parsed && Ci['R']==1 && Ci['G']==1 && Ci['B']==1 && Ci['A']==1)
                                                  //RGBA
                                                    Fill(StreamKind_Last, 0, "ColorSpace", "RGBA");
                                                else if ((Ci['Y']==1 && Ci['A']==1 && ((Ci['C']==1 && Ci['c']==1)
                                    //YCcA
                                                                                        || Ci['C']==2))
                                    //YCCA
                                                    || APP0_JFIF_Parsed
                                                //APP0 JFIF header present so YCCA
                                                    || (SamplingFactors[0].Ci==0 && SamplingFactors[1].Ci==1 && SamplingFactors[2].Ci==
& SamplingFactors[3].Ci==3)     //0123
                                                    || (SamplingFactors[0].Ci==1 && SamplingFactors[1].Ci==2 && SamplingFactors[2].Ci==
& SamplingFactors[3].Ci==4))    //1234
                                                    Fill(StreamKind_Last, 0, "ColorSpace", "YUVA");
                                                else if (APPE_Adobe0_transform==0)
                                                  //transform set to CMYK
                                                    Fill(StreamKind_Last, 0, "ColorSpace", "YCCB");
                                                break;
                                default:    ;
                            }
                            }
            }

            //Chroma subsampling
            if ((SamplingFactors.size()==3 || SamplingFactors.size()==4) && SamplingFactors[1].Hi==1 && SamplingFactors[2].Hi==1 && S
lingFactors[1].Vi==1 && SamplingFactors[2].Vi==1)
            {
                string ChromaSubsampling;
                switch (SamplingFactors[0].Hi)
                {
                    case 1 :
                            switch (SamplingFactors[0].Vi)
                            {
                                case 1 : ChromaSubsampling="4:4:4"; break;
                                default: ;
                            }
                            break;
                    case 2 :
                            switch (SamplingFactors[0].Vi)
                            {
                                case 1 : ChromaSubsampling="4:2:2"; break;
                                case 2 : ChromaSubsampling="4:2:0"; break;
                                default: ;
```

```
                            }
                            break;
                    case 4 :
                            switch (SamplingFactors[0].Vi)
                            {
                                case 1 : ChromaSubsampling="4:1:1"; break;
                                default: ;
                            }
                            break;
                    default: ;
                }
                if (!ChromaSubsampling.empty())
                {
                    if (SamplingFactors.size()==4)
                    {
                        if (ChromaSubsampling=="4:4:4" && SamplingFactors[3].Hi==1 && SamplingFactors[3].Vi==1)
                            ChromaSubsampling+=":4";
                        else
                            ChromaSubsampling+=":?";
                    }
                    Fill(StreamKind_Last, 0, "ChromaSubsampling", ChromaSubsampling);
                }
            }
        }
    FILLING_END();
}
```

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。

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