

## 原 XBMC源代码分析 4：视频播放器（dvdplayer）-解码器（以ffmpeg为例）

2014年01月07日 00:03:53 阅读数：9679

XBMC分析系列文章：

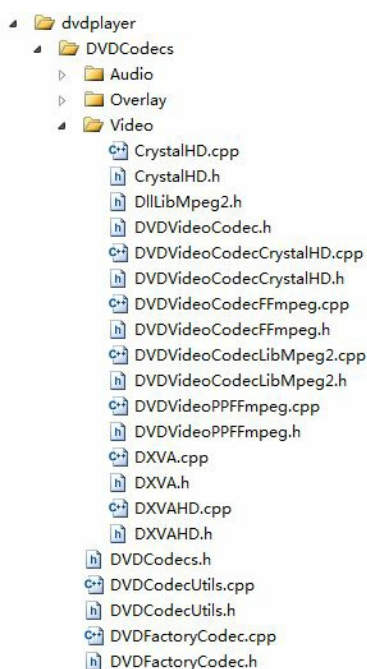
[XBMC源代码分析 1：整体结构以及编译方法](#)

[XBMC源代码分析 2：Addons（皮肤Skin）](#)

[XBMC源代码分析 3：核心部分（core）-综述](#)

本文我们分析XBMC中视频播放器（dvdplayer）中的解码器部分。由于解码器种类很多，不可能一一分析，因此以ffmpeg解码器为例进行分析。

XBMC解码器部分文件目录如下图所示：



解码器分为音频解码器和视频解码器。在这里我们看一下视频解码器中的FFMPEG解码器。对应DVDVideoCodecFFmpeg.h和DVDVideoCodecFFmpeg.cpp。

DVDVideoCodecFFmpeg.h源代码如下所示：

```
[cpp]
1.  /*
2.   * 雷霄骅
3.   *  leixiaohua1020@126.com
4.   *  中国传媒大学/数字电视技术
5.   *
6.   */
7.
8.  #include "DVDVideoCodec.h"
9.  #include "DVDResource.h"
10. #include "DllAvCodec.h"
11. #include "DllAvFormat.h"
12. #include "DllAvUtil.h"
13. #include "DllSwScale.h"
14. #include "DllAvFilter.h"
15. #include "DllPostProc.h"
16.
17. class CCriticalSection;
18. //封装的FFMPEG视频解码器
19. class CDVDVideoCodecFFmpeg : public CDVDVideoCodec
20. {
21. public:
22.     class IHardwareDecoder : public IDVDResourceCounted<IHardwareDecoder>
23.     {
24.     public:
25.         IHardwareDecoder() {}
26.         virtual ~IHardwareDecoder() {};
27.         virtual bool Open (AVCodecContext* avctx, const enum PixelFormat, unsigned int surfaces) = 0;
28.         virtual int Decode (AVCodecContext* avctx, AVFrame* frame) = 0;
29.         virtual bool GetPicture(AVCodecContext* avctx, AVFrame* frame, DVDVideoPicture* picture) = 0;
30.         virtual int Check (AVCodecContext* avctx) = 0;
```

```

30.     virtual int Check (AVCodecContext *avctx) = 0;
31.     virtual void Reset () {}
32.     virtual unsigned GetAllowedReferences() { return 0; }
33.     virtual const std::string Name() = 0;
34.     virtual CCriticalSection* Section() { return NULL; }
35. };
36.
37. CDVDVideoCodecFFmpeg();
38. virtual ~CDVDVideoCodecFFmpeg();
39. virtual bool Open(CDVDStreamInfo &hints, CDVDCodecOptions &options); //打开
40. virtual void Dispose(); //关闭
41. virtual int Decode(uint8_t* pData, int iSize, double dts, double pts); //解码
42. virtual void Reset();
43. bool GetPictureCommon(DVDVideoPicture* pDvdVideoPicture);
44. virtual bool GetPicture(DVDVideoPicture* pDvdVideoPicture);
45. virtual void SetDropState(bool bDrop);
46. virtual unsigned int SetFilters(unsigned int filters);
47. virtual const char* GetName() { return m_name.c_str(); }; // m_name is never changed after open
48. virtual unsigned GetConvergeCount();
49. virtual unsigned GetAllowedReferences();
50.
51. bool IsHardwareAllowed() { return !m_bSoftware; }
52. IHardwareDecoder * GetHardware() { return m_pHardware; };
53. void SetHardware(IHardwareDecoder* hardware)
54. {
55.     SAFE_RELEASE(m_pHardware);
56.     m_pHardware = hardware;
57.     UpdateName();
58. }
59.
60. protected:
61.     static enum PixelFormat GetFormat(struct AVCodecContext * avctx, const PixelFormat * fmt);
62.
63.     int FilterOpen(const CStdString& filters, bool scale);
64.     void FilterClose();
65.     int FilterProcess(AVFrame* frame);
66.
67.     void UpdateName()
68.     {
69.         if(m_pCodecContext->codec->name)
70.             m_name = CStdString("ff-") + m_pCodecContext->codec->name;
71.         else
72.             m_name = "ffmpeg";
73.
74.         if(m_pHardware)
75.             m_name += "-" + m_pHardware->Name();
76.     }
77.
78.     AVFrame* m_pFrame;
79.     AVCodecContext* m_pCodecContext;
80.
81.     CStdString m_filters;
82.     CStdString m_filters_next;
83.     AVFilterGraph* m_pFilterGraph;
84.     AVFilterContext* m_pFilterIn;
85.     AVFilterContext* m_pFilterOut;
86. #if defined(LIBAVFILTER_AVFRAME_BASED)
87.     AVFrame* m_pFilterFrame;
88. #else
89.     AVFilterBufferRef* m_pBufferRef;
90. #endif
91.
92.     int m_iPictureWidth;
93.     int m_iPictureHeight;
94.
95.     int m_iScreenWidth;
96.     int m_iScreenHeight;
97.     int m_iOrientation; // orientation of the video in degrees counter clockwise
98.
99.     unsigned int m_uSurfacesCount;
100.     //封装DLL的各种类
101.     DLLAvCodec m_dllAvCodec;
102.     DLLAvUtil m_dllAvUtil;
103.     DLLSwScale m_dllSwScale;
104.     DLLAvFilter m_dllAvFilter;
105.     DLLPostProc m_dllPostProc;
106.
107.     std::string m_name;
108.     bool m_bSoftware;
109.     bool m_isHi10p;
110.     IHardwareDecoder *m_pHardware;
111.     int m_iLastKeyframe;
112.     double m_dts;
113.     bool m_started;
114.     std::vector<PixelFormat> m_formats;
115. };

```

该类中以下几个函数包含了解码器的几种功能：

**virtual bool Open(CDVDStreamInfo &hints, CDVDCodecOptions &options); //打开**

```
virtual void Dispose();//关闭
```

```
virtual int Decode(uint8_t* pData, int iSize, double dts, double pts);//解码
```

```
virtual void Reset();//复位
```

为了说明这一点，我们可以看一下视频解码器中的libmpeg2解码器，对应DVDVideoCodecLibMpeg2.h。可以看出这几个函数是一样的。

DVDVideoCodecLibMpeg2.h源代码如下：

```
[cpp]
1.  /*
2.  * 雷霄骅
3.  *  leixiaohua1020@126.com
4.  *  中国传媒大学/数字电视技术
5.  *
6.  */
7.  #include "DVDVideoCodec.h"
8.  #include "DllLibMpeg2.h"
9.
10. class CDVDVideoCodecLibMpeg2 : public CDVDVideoCodec
11. {
12. public:
13.     CDVDVideoCodecLibMpeg2();
14.     virtual ~CDVDVideoCodecLibMpeg2();
15.     virtual bool Open(CDVDStreamInfo &hints, CDVDCodecOptions &options);
16.     virtual void Dispose();
17.     virtual int Decode(uint8_t* pData, int iSize, double dts, double pts);
18.     virtual void Reset();
19.     virtual bool GetPicture(DVDVideoPicture* pDvdVideoPicture);
20.     virtual bool GetUserData(DVDVideoUserData* pDvdVideoUserData);
21.
22.     virtual void SetDropState(bool bDrop);
23.     virtual const char* GetName() { return "libmpeg2"; }
24.
25. protected:
26.     DVDVideoPicture* GetBuffer(unsigned int width, unsigned int height);
27.     inline void ReleaseBuffer(DVDVideoPicture* pPic);
28.     inline void DeleteBuffer(DVDVideoPicture* pPic);
29.
30.     static int GuessAspect(const mpeg2_sequence_t *sequence, unsigned int *pixel_width, unsigned int *pixel_height);
31.
32.     mpeg2dec_t* m_pHandle;
33.     const mpeg2_info_t* m_pInfo;
34.     DllLibMpeg2 m_dll;
35.
36.     unsigned int m_irfffpattern;
37.     bool m_bFilm; //Signals that we have film material
38.     bool m_bIs422;
39.
40.     int m_hurry;
41.     double m_dts;
42.     double m_dts2;
43.     //The buffer of pictures we need
44.     DVDVideoPicture m_pVideoBuffer[3];
45.     DVDVideoPicture* m_pCurrentBuffer;
46. };
```

现在回到DVDVideoCodecFFmpeg.h。我们可以看一下上文所示的4个函数。

Open()

```
[cpp]
1.  //打开
2.  bool CDVDVideoCodecFFmpeg::Open(CDVDStreamInfo &hints, CDVDCodecOptions &options)
3.  {
4.      AVCodec* pCodec;
5.
6.      if(!m_dllAvUtil.Load()
7.      || !m_dllAvCodec.Load()
8.      || !m_dllSwScale.Load()
9.      || !m_dllPostProc.Load()
10.     || !m_dllAvFilter.Load())
11.         return false;
12.     //注册解码器
13.     m_dllAvCodec.avcodec_register_all();
14.     m_dllAvFilter.avfilter_register_all();
15.
16.     m_bSoftware = hints.software;
17.     m_iOrientation = hints.orientation;
18.
19.     for(std::vector<ERenderFormat>::iterator it = options.m_formats.begin(); it != options.m_formats.end(); ++it)
20.     {
21.         m_formats.push_back((PixelFormat)CDVDCodecUtils::PixfmtFromEFormat(*it));
22.         if(*it == RENDER_FMT_YUV420P)
23.             m_formats.push_back(PIX_FMT_YUVJ420P);
24.     }
25.     m_formats.push_back(PIX_FMT_NONE); /* always add none to get a terminated list in ffmpeg world */
26. }
```

```

27.     pCodec = NULL;
28.     m_pCodecContext = NULL;
29.
30.     if (hints.codec == AV_CODEC_ID_H264)
31.     {
32.         switch(hints.profile)
33.         {
34.             case FF_PROFILE_H264_HIGH_10:
35.             case FF_PROFILE_H264_HIGH_10_INTRA:
36.             case FF_PROFILE_H264_HIGH_422:
37.             case FF_PROFILE_H264_HIGH_422_INTRA:
38.             case FF_PROFILE_H264_HIGH_444_PREDICTIVE:
39.             case FF_PROFILE_H264_HIGH_444_INTRA:
40.             case FF_PROFILE_H264_CAVLC_444:
41.                 // this is needed to not open the decoders
42.                 m_bSoftware = true;
43.                 // this we need to enable multithreading for hi10p via advancedsettings
44.                 m_isHi10p = true;
45.                 break;
46.         }
47.     }
48.     //查找解码器
49.     if(pCodec == NULL)
50.         pCodec = m_dllAvCodec.avcodec_find_decoder(hints.codec);
51.
52.     if(pCodec == NULL)
53.     {
54.         CLog::Log(LOGDEBUG, "CDVDVideoCodecFFmpeg::Open() Unable to find codec %d", hints.codec);
55.         return false;
56.     }
57.
58.     CLog::Log(LOGNOTICE, "CDVDVideoCodecFFmpeg::Open() Using codec: %s", pCodec->long_name ? pCodec->long_name : pCodec->name);
59.
60.     if(m_pCodecContext == NULL)
61.         m_pCodecContext = m_dllAvCodec.avcodec_alloc_context3(pCodec);
62.
63.     m_pCodecContext->opaque = (void*)this;
64.     m_pCodecContext->debug_mv = 0;
65.     m_pCodecContext->debug = 0;
66.     m_pCodecContext->workaround_bugs = FF_BUG_AUTODETECT;
67.     m_pCodecContext->get_format = GetFormat;
68.     m_pCodecContext->codec_tag = hints.codec_tag;
69.     /* Only allow slice threading, since frame threading is more
70.      * sensitive to changes in frame sizes, and it causes crashes
71.      * during HW accel - so we unset it in this case.
72.      *
73.      * When we detect Hi10p and user did not disable hi10p multithreading
74.      * via advancedsettings.xml we keep the ffmpeg default thread type.
75.      */
76.     if(m_isHi10p && !g_advancedSettings.m_videoDisableHi10pMultithreading)
77.     {
78.         CLog::Log(LOGDEBUG, "CDVDVideoCodecFFmpeg::Open() Keep default threading for Hi10p: %d",
79.             m_pCodecContext->thread_type);
80.     }
81.     else if (CSettings::Get().GetBool("videoplayer.useframemtdec"))
82.     {
83.         CLog::Log(LOGDEBUG, "CDVDVideoCodecFFmpeg::Open() Keep default threading %d by videoplayer.useframemtdec",
84.             m_pCodecContext->thread_type);
85.     }
86.     else
87.         m_pCodecContext->thread_type = FF_THREAD_SLICE;
88.
89.     #if defined(TARGET_DARWIN_IOS)
90.         // ffmpeg with enabled neon will crash and burn if this is enabled
91.         m_pCodecContext->flags &= CODEC_FLAG_EMU_EDGE;
92.     #else
93.         if (pCodec->id != AV_CODEC_ID_H264 && pCodec->capabilities & CODEC_CAP_DR1
94.             && pCodec->id != AV_CODEC_ID_VP8
95.             )
96.             m_pCodecContext->flags |= CODEC_FLAG_EMU_EDGE;
97.     #endif
98.
99.     // if we don't do this, then some codecs seem to fail.
100.    m_pCodecContext->coded_height = hints.height;
101.    m_pCodecContext->coded_width = hints.width;
102.    m_pCodecContext->bits_per_coded_sample = hints.bitsperpixel;
103.
104.    if( hints.extradata && hints.extrasize > 0 )
105.    {
106.        m_pCodecContext->extradata_size = hints.extrasize;
107.        m_pCodecContext->extradata = (uint8_t*)m_dllAvUtil.av_mallocz(hints.extrasize + FF_INPUT_BUFFER_PADDING_SIZE);
108.        memcpy(m_pCodecContext->extradata, hints.extradata, hints.extrasize);
109.    }
110.
111.    // advanced setting override for skip loop filter (see avcodec.h for valid options)
112.    // TODO: allow per video setting?
113.    if (g_advancedSettings.m_iSkipLoopFilter != 0)
114.    {
115.        m_pCodecContext->skip_loop_filter = (AVDiscard)g_advancedSettings.m_iSkipLoopFilter;
116.    }
117.

```

```

118. // set any special options
119. for(std::vector<CDVDCodecOption>::iterator it = options.m_keys.begin(); it != options.m_keys.end(); ++it)
120. {
121.     if (it->m_name == "surfaces")
122.         m_uSurfacesCount = std::atoi(it->m_value.c_str());
123.     else
124.         m_dllAvUtil.av_opt_set(m_pCodecContext, it->m_name.c_str(), it->m_value.c_str(), 0);
125. }
126.
127. int num_threads = std::min(8 /*MAX_THREADS*/, g_cpuInfo.getCPUCount());
128. if( num_threads > 1 && !hints.software && m_pHardware == NULL // thumbnail extraction fails when run threaded
129. && ( pCodec->id == AV_CODEC_ID_H264
130.     || pCodec->id == AV_CODEC_ID_MPEG4 ))
131.     m_pCodecContext->thread_count = num_threads;
132. //打开解码器
133. if (m_dllAvCodec.avcodec_open2(m_pCodecContext, pCodec, NULL) < 0)
134. {
135.     CLog::Log(LOGDEBUG,"CDVDVideoCodecFFmpeg::Open() Unable to open codec");
136.     return false;
137. }
138. //初始化AVFrame
139. m_pFrame = m_dllAvCodec.avcodec_alloc_frame();
140. if (!m_pFrame) return false;
141.
142. #if defined(LIBAVFILTER_AVFRAME_BASED)
143.     m_pFilterFrame = m_dllAvUtil.av_frame_alloc();
144.     if (!m_pFilterFrame) return false;
145. #endif
146.
147.     UpdateName();
148.     return true;
149. }

```

#### Dispose()

```

1. //关闭
2. void CDVDVideoCodecFFmpeg::Dispose()
3. {
4.     //释放
5.     if (m_pFrame) m_dllAvUtil.av_free(m_pFrame);
6.     m_pFrame = NULL;
7.
8.     #if defined(LIBAVFILTER_AVFRAME_BASED)
9.         m_dllAvUtil.av_frame_free(&m_pFilterFrame);
10.    #endif
11.
12.    if (m_pCodecContext)
13.    {
14.        //关闭解码器
15.        if (m_pCodecContext->codec) m_dllAvCodec.avcodec_close(m_pCodecContext);
16.        if (m_pCodecContext->extradata)
17.        {
18.            m_dllAvUtil.av_free(m_pCodecContext->extradata);
19.            m_pCodecContext->extradata = NULL;
20.            m_pCodecContext->extradata_size = 0;
21.        }
22.        m_dllAvUtil.av_free(m_pCodecContext);
23.        m_pCodecContext = NULL;
24.    }
25.    SAFE_RELEASE(m_pHardware);
26.
27.    FilterClose();
28.
29.    m_dllAvCodec.Unload();
30.    m_dllAvUtil.Unload();
31.    m_dllAvFilter.Unload();
32.    m_dllPostProc.Unload();
33. }

```

#### Decode()

```

1. //解码
2. int CDVDVideoCodecFFmpeg::Decode(uint8_t* pData, int iSize, double dts, double pts)
3. {
4.     int iGotPicture = 0, len = 0;
5.
6.     if (!m_pCodecContext)
7.         return VC_ERROR;
8.
9.     if(pData)
10.         m_iLastKeyframe++;
11.
12.     shared_ptr<CSingleLock> lock;
13.     if(m_pHardware)

```



```

14. {
15.     CCriticalSection* section = m_pHardware->Section();
16.     if(section)
17.         lock = shared_ptr<CSingleLock>(new CSingleLock(*section));
18.
19.     int result;
20.     if(pData)
21.         result = m_pHardware->Check(m_pCodecContext);
22.     else
23.         result = m_pHardware->Decode(m_pCodecContext, NULL);
24.
25.     if(result)
26.         return result;
27. }
28.
29. if(m_pFilterGraph)
30. {
31.     int result = 0;
32.     if(pData == NULL)
33.         result = FilterProcess(NULL);
34.     if(result)
35.         return result;
36. }
37.
38. m_dts = dts;
39. m_pCodecContext->reordered_opaque = pts_dtoi(pts);
40. //初始化AVPacket
41. AVPacket avpkt;
42. m_dllAvCodec.av_init_packet(&avpkt);
43. avpkt.data = pData;
44. avpkt.size = iSize;
45. /* We lie, but this flag is only used by pngdec.c.
46.  * Setting it correctly would allow CorePNG decoding. */
47. avpkt.flags = AV_PKT_FLAG_KEY;
48. //解码
49. len = m_dllAvCodec.avcodec_decode_video2(m_pCodecContext, m_pFrame, &iGotPicture, &avpkt);
50.
51. if(m_iLastKeyframe < m_pCodecContext->has_b_frames + 2)
52.     m_iLastKeyframe = m_pCodecContext->has_b_frames + 2;
53.
54. if (len < 0)
55. {
56.     CLog::Log(LOGERROR, "%s - avcodec_decode_video returned failure", __FUNCTION__);
57.     return VC_ERROR;
58. }
59.
60. if (!iGotPicture)
61.     return VC_BUFFER;
62.
63. if(m_pFrame->key_frame)
64. {
65.     m_started = true;
66.     m_iLastKeyframe = m_pCodecContext->has_b_frames + 2;
67. }
68.
69. /* put a limit on convergence count to avoid huge mem usage on streams without keyframes */
70. if(m_iLastKeyframe > 300)
71.     m_iLastKeyframe = 300;
72.
73. /* h264 doesn't always have keyframes + won't output before first keyframe anyway */
74. if(m_pCodecContext->codec_id == AV_CODEC_ID_H264
75. || m_pCodecContext->codec_id == AV_CODEC_ID_SVQ3)
76.     m_started = true;
77.
78. if(m_pHardware == NULL)
79. {
80.     bool need_scale = std::find( m_formats.begin()
81.                                , m_formats.end()
82.                                , m_pCodecContext->pix_fmt) == m_formats.end();
83.
84.     bool need_reopen = false;
85.     if(!m_filters.Equals(m_filters_next))
86.         need_reopen = true;
87.
88.     if(m_pFilterIn)
89.     {
90.         if(m_pFilterIn->outputs[0]->format != m_pCodecContext->pix_fmt
91.         || m_pFilterIn->outputs[0]->w      != m_pCodecContext->width
92.         || m_pFilterIn->outputs[0]->h      != m_pCodecContext->height)
93.             need_reopen = true;
94.     }
95.
96.     // try to setup new filters
97.     if (need_reopen || (need_scale && m_pFilterGraph == NULL))
98.     {
99.         m_filters = m_filters_next;
100.
101.         if(FilterOpen(m_filters, need_scale) < 0)
102.             FilterClose();
103.     }
104. }

```

```
105.
106.     int result;
107.     if(m_pHardware)
108.         result = m_pHardware->Decode(m_pCodecContext, m_pFrame);
109.     else if(m_pFilterGraph)
110.         result = FilterProcess(m_pFrame);
111.     else
112.         result = VC_PICTURE | VC_BUFFER;
113.
114.     if(result & VC_FLUSHED)
115.         Reset();
116.
117.     return result;
118. }
```

Reset()

```
[cpp]  
1. //复位
2. void CDVDVideoCodecFFmpeg::Reset()
3. {
4.     m_started = false;
5.     m_iLastKeyframe = m_pCodecContext->has_b_frames;
6.     m_dllAvCodec.avcodec_flush_buffers(m_pCodecContext);
7.
8.     if (m_pHardware)
9.         m_pHardware->Reset();
10.
11.     m_filters = "";
12.     FilterClose();
13. }
```

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/17512509>

文章标签：[xbmc](#) [源代码](#) [ffmpeg](#) [媒体中心](#) [播放器](#)

个人分类：[XBMC](#) [FFMPEG](#)

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