原 ffdshow 源代码分析 6： 对解码器的dll的封装（libavcodec）

2013年11月12日 00:09:11　阅读数：7714

===

ffdshow源代码分析系列文章列表：

===

ffdshow封装了多个视音频解码器，比如libmpeg2，libavcodec，xvid等等。其中最重要的是libavcodec，这个是ffmpeg提供的解码器，在ffdshow中起到了"挑大梁"的作用。本文分析ffdshow对解码器的封装方式，就以libavcodec为例。

在ffdshow中，libavcodec的被封装在ffmpeg.dll文件中，通过加载该dll中的函数，就可以使用libavcodec的各种方法。

Ffmpeg对libavcodec的封装类的定义位于codecs->libavcodec->Tlibavcodec.h。实现则位于codecs->libavcodec->Tlibavcodec.cpp。

先来看一看Tlibavcodec.h：

[cpp]

```cpp
/*
 *雷霄骅
 *leixiaohua1020@126.com
 *中国传媒大学/数字电视技术
 */
#ifndef _TLIBAVCODEC_H_
#define _TLIBAVCODEC_H_
//将FFmpeg的Dll中的方法封装到一个类中，以供使用
#include "../codecs/ffcodecs.h"
#include <dxva.h>
#include "TpostprocSettings.h"
#include "ffImgfmt.h"
#include "libavfilter/vf_yadif.h"
#include "libavfilter/gradfun.h"
#include "libswscale/swscale.h"

struct AVCodecContext;
struct AVCodec;
struct AVFrame;
struct AVPacket;
struct AVCodecParserContext;
struct SwsContext;
struct SwsParams;
struct PPMode;
struct AVDictionary;

struct Tconfig;
class Tdll;
struct DSPContext;
struct TlibavcodecExt;
//封装FFMPEG
//里面的函数基本上是FFMPEG的API
struct Tlibavcodec {
private:
    int (*libswscale_sws_scale)(struct SwsContext *context, const uint8_t* const srcSlice[], const int srcStride[],
                                int srcSliceY, int srcSliceH, uint8_t* const dst[], const int dstStride[]);
                                int srcSliceY, int srcSliceH, uint8_t* const dst[], const int dstStride[]);
    //加载DLL的类
    Tdll *dll;
    int refcount;
    static int get_buffer(AVCodecContext *c, AVFrame *pic);
    CCritSec csOpenClose;
public:
    Tlibavcodec(const Tconfig *config);
    ~Tlibavcodec();
    static void avlog(AVCodecContext*, int, const char*, va_list);
    static void avlogMsgBox(AVCodecContext*, int, const char*, va_list);
    void AddRef(void) {
        refcount++;
    }
    void Release(void) {
        if (--refcount < 0) {
            delete this;
        }
    }
    static bool getVersion(const Tconfig *config, ffstring &vers, ffstring &license);
    static bool check(const Tconfig *config);
    static int ppCpuCaps(uint64_t csp);
    static void pp_mode_defaults(PPMode &ppMode);
    static int getPPmode(const TpostprocSettings *cfg, int currentq);
    static void swsInitParams(SwsParams *params, int resizeMethod);
    static void swsInitParams(SwsParams *params, int resizeMethod, int flags);

    bool ok;
    AVCodecContext* avcodec_alloc_context(AVCodec *codec, TlibavcodecExt *ext = NULL);

    void (*avcodec_register_all)(void);
    AVCodecContext* (*avcodec_alloc_context0)(AVCodec *codec);
    AVCodec* (*avcodec_find_decoder)(AVCodecID codecId);
    AVCodec* (*avcodec_find_encoder)(AVCodecID id);
    int (*avcodec_open0)(AVCodecContext *avctx, AVCodec *codec, AVDictionary **options);
    int  avcodec_open(AVCodecContext *avctx, AVCodec *codec);
    AVFrame* (*avcodec_alloc_frame)(void);
    int (*avcodec_decode_video2)(AVCodecContext *avctx, AVFrame *picture,
                                 int *got_picture_ptr,
                                 AVPacket *avpkt);
    int (*avcodec_decode_audio3)(AVCodecContext *avctx, int16_t *samples,
                                 int *frame_size_ptr,
                                 AVPacket *avpkt);
    int (*avcodec_encode_video)(AVCodecContext *avctx, uint8_t *buf, int buf_size, const AVFrame *pict);
    int (*avcodec_encode_audio)(AVCodecContext *avctx, uint8_t *buf, int buf_size, const short *samples);
    void (*avcodec_flush_buffers)(AVCodecContext *avctx);
    int (*avcodec_close0)(AVCodecContext *avctx);
    int  avcodec_close(AVCodecContext *avctx);

    void (*av_log_set_callback)(void (*)(AVCodecContext*, int, const char*, va_list));
    void* (*av_log_get_callback)(void);
    int (*av_log_get_level)(void);
    void (*av_log_set_level)(int);

    void (*av_set_cpu_flags_mask)(int mask);

    int (*avcodec_default_get_buffer)(AVCodecContext *s, AVFrame *pic);
    void (*avcodec_default_release_buffer)(AVCodecContext *s, AVFrame *pic);
    int (*avcodec_default_reget_buffer)(AVCodecContext *s, AVFrame *pic);
    const char* (*avcodec_get_current_idct)(AVCodecContext *avctx);
    void (*avcodec_get_encoder_info)(AVCodecContext *avctx, int *xvid_build, int *divx_version, int *divx_build, int *lavc_build);

    void* (*av_mallocz)(size_t size);
    void (*av_free)(void *ptr);

    AVCodecParserContext* (*av_parser_init)(int codec_id);
    int (*av_parser_parse2)(AVCodecParserContext *s, AVCodecContext *avctx, uint8_t **poutbuf, int *poutbuf_size, const uint8_t *buf, int buf_size, int64_t pts, int64_t dts, int64_t pos);
    void (*av_parser_close)(AVCodecParserContext *s);

    void (*av_init_packet)(AVPacket *pkt);
    uint8_t* (*av_packet_new_side_data)(AVPacket *pkt, enum AVPacketSideDataType type, int size);

    int (*avcodec_h264_search_recovery_point)(AVCodecContext *avctx,
            const uint8_t *buf, int buf_size, int *recovery_frame_cnt);

    static const char_t *idctNames[], *errorRecognitions[], *errorConcealments[];
    struct Tdia_size {
        int size;
        const char_t *descr;
    };
    static const Tdia_size dia_sizes[];

    //libswscale imports
    SwsContext* (*sws_getCachedContext)(struct SwsContext *context, int srcW, int srcH, enum PixelFormat srcFormat,
                                        int dstW, int dstH, enum PixelFormat dstFormat, int flags,
                                        SwsFilter *srcFilter, SwsFilter *dstFilter, const double *param, SwsParams *ffdshow_params);

    void (*sws_freeContext)(SwsContext *c);
    SwsFilter* (*sws_getDefaultFilter)(float lumaGBlur, float chromaGBlur,
                                       float lumaSharpen, float chromaSharpen,
                                       float chromaHShift, float chromaVShift,
                                       int verbose);
    void (*sws_freeFilter)(SwsFilter *filter);
    int sws_scale(struct SwsContext *context, const uint8_t* const srcSlice[], const stride_t srcStride[],
                  int srcSliceY, int srcSliceH, uint8_t* const dst[], const stride_t dstStride[]);
    SwsVector *(*sws_getConstVec)(double c, int length);
    SwsVector *(*sws_getGaussianVec)(double variance, double quality);
    void (*sws_normalizeVec)(SwsVector *a, double height);
    void (*sws_freeVec)(SwsVector *a);
    int (*sws_setColorspaceDetails)(struct SwsContext *c, const int inv_table[4],
                                    int srcRange, const int table[4], int dstRange,
                                    int brightness, int contrast, int saturation);
    const int* (*sws_getCoefficients)(int colorspace);

    int (*GetCPUCount)(void);

    //libpostproc imports
    void (*pp_postprocess)(const uint8_t * src[3], const stride_t srcStride[3],
                           uint8_t * dst[3], const stride_t dstStride[3],
                           int horizontalSize, int verticalSize,
                           const /*QP_STORE_T*/int8_t *QP_store,  int QP_stride,
                           /*pp_mode*/void *mode, /*pp_context*/void *ppContext, int pict_type);
    /*pp_context*/
    void *(*pp_get_context)(int width, int height, int flags);
    void (*pp_free_context)(/*pp_context*/void *ppContext);
    void (*ff_simple_idct_mmx)(int16_t *block);

    // DXVA imports
    int (*av_h264_decode_frame)(struct AVCodecContext* avctx, uint8_t *buf, int buf_size);
    int (*av_vc1_decode_frame)(struct AVCodecContext* avctx, uint8_t *buf, int buf_size);

    // === H264 functions
    int (*FFH264CheckCompatibility)(int nWidth, int nHeight, struct AVCodecContext* pAVCtx, BYTE* pBuffer, UINT nSize, int nPCIVendor, int nPCIDevice, LARGE_INTEGER VideoDriverVersion);
    int (*FFH264DecodeBuffer)(struct AVCodecContext* pAVCtx, BYTE* pBuffer, UINT nSize, int* pFramePOC, int* pOutPOC, REFERENCE_TIME* pOutrtStart);
    HRESULT(*FFH264BuildPicParams)(DXVA_PicParams_H264* pDXVAPicParams, DXVA_Qmatrix_H264* pDXVAScalingMatrix, int* nFieldType, int* nSliceType, struct AVCodecContext* pAVCtx, int nPCIVendor);

    void (*FFH264SetCurrentPicture)(int nIndex, DXVA_PicParams_H264* pDXVAPicParams, struct AVCodecContext* pAVCtx);
    void (*FFH264UpdateRefFramesList)(DXVA_PicParams_H264* pDXVAPicParams, struct AVCodecContext* pAVCtx);
    BOOL (*FFH264IsRefFrameInUse)(int nFrameNum, struct AVCodecContext* pAVCtx);
    void (*FF264UpdateRefFrameSliceLong)(DXVA_PicParams_H264* pDXVAPicParams, DXVA_Slice_H264_Long* pSlice, struct AVCodecContext* pAVCtx);
    void (*FFH264SetDxvaSliceLong)(struct AVCodecContext* pAVCtx, void* pSliceLong);

    // === VC1 functions
    HRESULT(*FFVC1UpdatePictureParam)(DXVA_PictureParameters* pPicParams, struct AVCodecContext* pAVCtx, int* nFieldType, int* nSliceType, BYTE* pBuffer, UINT nSize, UINT* nFrameSize, BOOL b_SecondField, BOOL* b_repeat_pict);
    int (*FFIsSkipped)(struct AVCodecContext* pAVCtx);

    // === Common functions
    char*    (*GetFFMpegPictureType)(int nType);
    unsigned long(*FFGetMBNumber)(struct AVCodecContext* pAVCtx);

    // yadif
    void (*yadif_init)(YADIFContext *yadctx);
    void (*yadif_uninit)(YADIFContext *yadctx);
    void (*yadif_filter)(YADIFContext *yadctx, uint8_t *dst[3], stride_t dst_stride[3], int width, int height, int parity, int tff);


    // gradfun
    int (*gradfunInit)(GradFunContext *ctx, const char *args, void *opaque);
    void (*gradfunFilter)(GradFunContext *ctx, uint8_t *dst, uint8_t *src, int width, int height, int dst_linesize, int src_linesize, int r);
};

#endif
```

从Tlibavcodec定义可以看出，里面包含了大量的ffmpeg中的API，占据了很大的篇幅。通过调用这些API，就可以使用livavcodec的各种功能。

在Tlibavcodec的定义中，有一个变量：Tdll *dll，通过该变量，就可以加载ffmpeg.dll中的方法。

先来看一下Tdll的定义：

```cpp
/*
 *雷霄骅
 *leixiaohua1020@126.com
 *中国传媒大学/数字电视技术
 */
 #ifndef _TDLL_H_
#define _TDLL_H_

#include "Tconfig.h"
//操作Dll的类
class Tdll
```

```
{
public:
    bool ok;
    Tdll(const char_t *dllName1, const Tconfig *config, bool explicitFullPath = false) {
        char_t name[MAX_PATH], ext[MAX_PATH];
        _splitpath_s(dllName1, NULL, 0, NULL, 0, name, countof(name), ext, countof(ext));
        if (config && !explicitFullPath) {
            char_t dllName2[MAX_PATH]; //installdir+filename+ext
            _makepath_s(dllName2, countof(dllName2), NULL, config->pth, name, ext);
            hdll = LoadLibrary(dllName2);
        } else {
            hdll = NULL;
        }
        if (!hdll) {
            hdll = LoadLibrary(dllName1);
            if (!hdll && !explicitFullPath) {
                if (config) {
                    char_t dllName3[MAX_PATH]; //ffdshow.ax_path+filename+ext
                    _makepath_s(dllName3, countof(dllName3), NULL, config->epth, name, ext);
                    hdll = LoadLibrary(dllName3);
                }
                if (!hdll) {
                    char_t dllName0[MAX_PATH]; //only filename+ext - let Windows find it
                    _makepath_s(dllName0, countof(dllName0), NULL, NULL, name, ext);
                    hdll = LoadLibrary(dllName0);
                }
            }
        }
        ok = (hdll != NULL);
    }
    ~Tdll() {
        if (hdll) {
            FreeLibrary(hdll);
        }
    }
    HMODULE hdll;
    //封装一下直接加载Dll的GetProcAddress
    template<class T> __forceinline void loadFunction(T &fnc, const char *name) {
        fnc = hdll ? (T)GetProcAddress(hdll, name) : NULL;
        ok &= (fnc != NULL);
    }
    template<class T> __forceinline void loadFunctionByIndex(T &fnc, uint16_t id) {
        uint32_t id32 = uint32_t(id);
        fnc = hdll ?
              (T) GetProcAddress(hdll, (LPCSTR)id32) :
              NULL;
        ok &= (fnc != NULL);
    }
    //检查Dll的状态是否正常
    static bool check(const char_t *dllName1, const Tconfig *config) {
        char_t name[MAX_PATH], ext[MAX_PATH];
        _splitpath_s(dllName1, NULL, 0, NULL, 0, name, countof(name), ext, countof(ext));
        if (config) {
            char_t dllName2[MAX_PATH]; //installdir+filename+ext
            _makepath_s(dllName2, countof(dllName2), NULL, config->pth, name, ext);
            if (fileexists(dllName2)) {
                return true;
            }
        }
        if (fileexists(dllName1)) {
            return true;
        }
        if (config) {
            char_t dllName3[MAX_PATH]; //ffdshow.ax_path+filename+ext
            _makepath_s(dllName3, MAX_PATH, NULL, config->epth, name, ext);
            if (fileexists(dllName3)) {
                return true;
            }
        }
        char_t dllName0[MAX_PATH]; //only filename+ext - let Windows find it
        _makepath_s(dllName0, countof(dllName0), NULL, NULL, name, ext);
        char_t dir0[MAX_PATH], *dir0flnm;
        if (SearchPath(NULL, dllName0, NULL, MAX_PATH, dir0, &dir0flnm)) {
            return true;
        }
        return false;
    }
};

#endif
```

从Tdll的定义可以看出，该类的loadFunction()函数封装了系统使用Dll功能的函数GetProcAddress()。

该类的构造函数Tdll()封装了系统加载Dll的函数LoadLibrary()。

此外该类还提供了check()用于检查Dll。

对于Tdll的分析先告一段落，现在我们回到Tlibavcodec，来看看它是如何加载libavcodec的函数的。查看一下Tlibavcodec的类的实现，位于codecs->libavcodec->Tlibavcodec.cpp。

该类的实现代码比较长，因此只能选择重要的函数查看一下。首先来看一下构造函数：

[cpp]

```cpp
//===================================== Tlibavcodec ====================================
//FFMPEG封装类的构造函数
Tlibavcodec::Tlibavcodec(const Tconfig *config): refcount(0)
{
    //加载FFMPEG的Dll
    dll = new Tdll(_l("ffmpeg.dll"), config);
    //加载各个函数
    dll->loadFunction(avcodec_register_all, "avcodec_register_all");
    dll->loadFunction(avcodec_find_decoder, "avcodec_find_decoder");
    dll->loadFunction(avcodec_open0, "avcodec_open2");
    dll->loadFunction(avcodec_alloc_context0, "avcodec_alloc_context3");
    dll->loadFunction(avcodec_alloc_frame, "avcodec_alloc_frame");
    dll->loadFunction(avcodec_decode_video2, "avcodec_decode_video2");
    dll->loadFunction(avcodec_flush_buffers, "avcodec_flush_buffers");
    dll->loadFunction(avcodec_close0, "avcodec_close");
    dll->loadFunction(av_log_set_callback, "av_log_set_callback");
    dll->loadFunction(av_log_get_callback, "av_log_get_callback");
    dll->loadFunction(av_log_get_level, "av_log_get_level");
    dll->loadFunction(av_log_set_level, "av_log_set_level");
    dll->loadFunction(av_set_cpu_flags_mask, "av_set_cpu_flags_mask");
    dll->loadFunction(av_mallocz, "av_mallocz");
    dll->loadFunction(av_free, "av_free");
    dll->loadFunction(avcodec_default_get_buffer, "avcodec_default_get_buffer");
    dll->loadFunction(avcodec_default_release_buffer, "avcodec_default_release_buffer");
    dll->loadFunction(avcodec_default_reget_buffer, "avcodec_default_reget_buffer");
    dll->loadFunction(avcodec_get_current_idct, "avcodec_get_current_idct");
    dll->loadFunction(avcodec_get_encoder_info, "avcodec_get_encoder_info");
    dll->loadFunction(av_init_packet, "av_init_packet");
    dll->loadFunction(av_packet_new_side_data, "av_packet_new_side_data");
    dll->loadFunction(avcodec_h264_search_recovery_point, "avcodec_h264_search_recovery_point");

    dll->loadFunction(avcodec_decode_audio3, "avcodec_decode_audio3");

    dll->loadFunction(avcodec_find_encoder, "avcodec_find_encoder");
    dll->loadFunction(avcodec_encode_video, "avcodec_encode_video");
    dll->loadFunction(avcodec_encode_audio, "avcodec_encode_audio");

    dll->loadFunction(av_parser_init, "av_parser_init");
    dll->loadFunction(av_parser_parse2, "av_parser_parse2");
    dll->loadFunction(av_parser_close, "av_parser_close");

    //libswscale methods
    dll->loadFunction(sws_getCachedContext, "sws_getCachedContext");
    dll->loadFunction(sws_freeContext, "sws_freeContext");
    dll->loadFunction(sws_getDefaultFilter, "sws_getDefaultFilter");
    dll->loadFunction(sws_freeFilter, "sws_freeFilter");
    dll->loadFunction(libswscale_sws_scale, "sws_scale");

    dll->loadFunction(GetCPUCount, "GetCPUCount");
    dll->loadFunction(sws_getConstVec, "sws_getConstVec");
    dll->loadFunction(sws_getGaussianVec, "sws_getGaussianVec");
    dll->loadFunction(sws_normalizeVec, "sws_normalizeVec");
    dll->loadFunction(sws_freeVec, "sws_freeVec");
    dll->loadFunction(sws_setColorspaceDetails, "sws_setColorspaceDetails");
    dll->loadFunction(sws_getCoefficients, "sws_getCoefficients");

    //libpostproc methods
    dll->loadFunction(pp_postprocess, "pp_postprocess");
    dll->loadFunction(pp_get_context, "pp_get_context");
    dll->loadFunction(pp_free_context, "pp_free_context");
    dll->loadFunction(ff_simple_idct_mmx, "ff_simple_idct_mmx");

    //DXVA methods
    dll->loadFunction(av_h264_decode_frame, "av_h264_decode_frame");
    dll->loadFunction(av_vc1_decode_frame, "av_vc1_decode_frame");

    dll->loadFunction(FFH264CheckCompatibility, "FFH264CheckCompatibility");
    dll->loadFunction(FFH264DecodeBuffer, "FFH264DecodeBuffer");
    dll->loadFunction(FFH264BuildPicParams, "FFH264BuildPicParams");
    dll->loadFunction(FFH264SetCurrentPicture, "FFH264SetCurrentPicture");
    dll->loadFunction(FFH264UpdateRefFramesList, "FFH264UpdateRefFramesList");
    dll->loadFunction(FFH264IsRefFrameInUse, "FFH264IsRefFrameInUse");
    dll->loadFunction(FF264UpdateRefFrameSliceLong, "FF264UpdateRefFrameSliceLong");
    dll->loadFunction(FFH264SetDxvaSliceLong, "FFH264SetDxvaSliceLong");

    dll->loadFunction(FFVC1UpdatePictureParam, "FFVC1UpdatePictureParam");
    dll->loadFunction(FFIsSkipped, "FFIsSkipped");

    dll->loadFunction(GetFFMpegPictureType, "GetFFMpegPictureType");
    dll->loadFunction(FFGetMBNumber, "FFGetMBNumber");

    //yadif methods
    dll->loadFunction(yadif_init, "yadif_init");
    dll->loadFunction(yadif_uninit, "yadif_uninit");
    dll->loadFunction(yadif_filter, "yadif_filter");

    //gradfun
    dll->loadFunction(gradfunInit, "gradfunInit");
    dll->loadFunction(gradfunFilter, "gradfunFilter");

    ok = dll->ok;
    //加载完毕后，进行注册
    if (ok) {
        avcodec_register_all();
        av_log_set_callback(avlog);
    }
}
```

该构造函数尽管篇幅比较长，但是还是比较好理解的，主要完成了3步：

1. 创建一个Tdll类的对象，加载“ffmpeg.dll”。
2. 使用loadFunction()加载各种函数。
3. 最后调用avcodec_register_all()注册各种解码器。

Tlibavcodec的析构函数则比较简单：

[cpp]

```cpp
Tlibavcodec::~Tlibavcodec()
{
    delete dll;
}
```

检查Dll的函数也比较简单：

[cpp]

```cpp
bool Tlibavcodec::check(const Tconfig *config)
{
    return Tdll::check(_l("ffmpeg.dll"), config);
}
```

此外，可能是出于某些功能的考虑，ffdshow还自己写了几个函数，但是限于篇幅不能一一介绍，在这里只介绍一个：

获取libavcodec版本：

[cpp]

```cpp
bool Tlibavcodec::getVersion(const Tconfig *config, ffstring &vers, ffstring &license)
{
    Tdll *dl = new Tdll(_l("ffmpeg.dll"), config);

    void (*av_getVersion)(char **version, char **build, char **datetime, const char* *license);
    dl->loadFunction(av_getVersion, "getVersion");
    bool res;
    if (av_getVersion) {
        res = true;
        char *version, *build, *datetime;
        const char *lic;
        av_getVersion(&version, &build, &datetime, &lic);
        vers = (const char_t*)text<char_t>(version) + ffstring(_l(" (")) + (const char_t*)text<char_t>(datetime) + _l(")");
        license = text<char_t>(lic);
    } else {
        res = false;
        vers.clear();
        license.clear();
    }
    delete dl;
    return res;
}
```

文章标签： ffdshow ffmpeg libavcodec dll 解码器

个人分类： ffdshow

所属专栏： 开源多媒体项目源代码分析