

## ffds show 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：7704

ffds show源代码分析系列文章列表：

[ffds show 源代码分析 1：整体结构](#)

[ffds show 源代码分析 2：位图覆盖滤镜（对话框部分Dialog）](#)

[ffds show 源代码分析 3：位图覆盖滤镜（设置部分Settings）](#)

[ffds show 源代码分析 4：位图覆盖滤镜（滤镜部分Filter）](#)

[ffds show 源代码分析 5：位图覆盖滤镜（总结）](#)

[ffds show 源代码分析 6：对解码器的dll的封装（libavcodec）](#)

[ffds show 源代码分析 7：libavcodec视频解码器类（TvideoCodecLibavcodec）](#)

[ffds show 源代码分析 8：视频解码器类（TvideoCodecDec）](#)

[ffds show 源代码分析 9：编解码器有关类的总结](#)



ffds show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ffds show中起到了“挑大梁”的作用。本文分析ffds show对解码器的封装方式，就以libavcodec为例。

在ffds show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。

Fmpeg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]
1.  /*
2.  *雷霄骅
3.  *leixiaohua1020@126.com
4.  *中国传媒大学/数字电视技术
5.  */
6.  #ifndef _TLIBAVCODEC_H_
7.  #define _TLIBAVCODEC_H_
8.  //将FFmpeg的DLL中的方法封装到一个类中，以供使用
9.  #include "../codecs/ffcodecs.h"
10. #include <dxva.h>
11. #include "TpostprocSettings.h"
12. #include "ffimgfmt.h"
13. #include "libavfilter/vf_yadif.h"
14. #include "libavfilter/gradfun.h"
15. #include "libswscale/swscale.h"
16.
17. struct AVCodecContext;
18. struct AVCodec;
19. struct AVFrame;
20. struct AVPacket;
21. struct AVCodecParserContext;
22. struct SwsContext;
23. struct SwsParams;
24. struct PPMODE;
25. struct AVDictionary;
26.
27. struct Tconfig;
28. class Tdll;
29. struct DSPContext;
30. struct TlibavcodecExt;
31. //封装FFMPEG
32. //里面的函数基本上是FFMPEG的API
33. struct Tlibavcodec {
34. private:
35.     int (*libswscale_sws_scale)(struct SwsContext *context, const uint8_t* const srcSlice[], const int srcStride[],
36.                                int srcSliceY, int srcSliceH, uint8_t* const dst[], const int dstStride[]);
```

```

37. //加载DLL的类
38. Tdll *dll;
39. int refcount;
40. static int get_buffer(AVCodecContext *c, AVFrame *pic);
41. CCritSec csOpenClose;
42. public:
43. Tlibavcodec(const Tconfig *config);
44. ~Tlibavcodec();
45. static void avlog(AVCodecContext*, int, const char*, va_list);
46. static void avlogMsgBox(AVCodecContext*, int, const char*, va_list);
47. void AddRef(void) {
48.     refcount++;
49. }
50. void Release(void) {
51.     if (--refcount < 0) {
52.         delete this;
53.     }
54. }
55. static bool getVersion(const Tconfig *config, ffstring &vers, ffstring &license);
56. static bool check(const Tconfig *config);
57. static int ppCpuCaps(uint64_t csp);
58. static void pp_mode_defaults(PPMode &ppMode);
59. static int getPPmode(const TpostprocSettings *cfg, int currentq);
60. static void swsInitParams(SwsParams *params, int resizeMode);
61. static void swsInitParams(SwsParams *params, int resizeMode, int flags);
62.
63. bool ok;
64. AVCodecContext* avcodec_alloc_context(AVCodec *codec, TlibavcodecExt *ext = NULL);
65.
66. void (*avcodec_register_all)(void);
67. AVCodecContext* (*avcodec_alloc_context0)(AVCodec *codec);
68. AVCodec* (*avcodec_find_decoder)(AVCodecID codecId);
69. AVCodec* (*avcodec_find_encoder)(AVCodecID id);
70. int (*avcodec_open0)(AVCodecContext *avctx, AVCodec *codec, AVDictionary **options);
71. int avcodec_open(AVCodecContext *avctx, AVCodec *codec);
72. AVFrame* (*avcodec_alloc_frame)(void);
73. int (*avcodec_decode_video2)(AVCodecContext *avctx, AVFrame *picture,
74.     int *got_picture_ptr,
75.     AVPacket *avpkt);
76. int (*avcodec_decode_audio3)(AVCodecContext *avctx, int16_t *samples,
77.     int *frame_size_ptr,
78.     AVPacket *avpkt);
79. int (*avcodec_encode_video)(AVCodecContext *avctx, uint8_t *buf, int buf_size, const AVFrame *pict);
80. int (*avcodec_encode_audio)(AVCodecContext *avctx, uint8_t *buf, int buf_size, const short *samples);
81. void (*avcodec_flush_buffers)(AVCodecContext *avctx);
82. int (*avcodec_close0)(AVCodecContext *avctx);
83. int avcodec_close(AVCodecContext *avctx);
84.
85. void (*av_log_set_callback)(void (*)(AVCodecContext*, int, const char*, va_list));
86. void* (*av_log_get_callback)(void);
87. int (*av_log_get_level)(void);
88. void (*av_log_set_level)(int);
89.
90. void (*av_set_cpu_flags_mask)(int mask);
91.
92. int (*avcodec_default_get_buffer)(AVCodecContext *s, AVFrame *pic);
93. void (*avcodec_default_release_buffer)(AVCodecContext *s, AVFrame *pic);
94. int (*avcodec_default_reget_buffer)(AVCodecContext *s, AVFrame *pic);
95. const char* (*avcodec_get_current_idct)(AVCodecContext *avctx);
96. void (*avcodec_get_encoder_info)(AVCodecContext *avctx, int *xvid_build, int *divx_version, int *divx_build, int *lavc_build);
97.
98. void* (*av_mallocz)(size_t size);
99. void (*av_free)(void *ptr);
100.
101. AVCodecParserContext* (*av_parser_init)(int codec_id);
102. 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);
103. void (*av_parser_close)(AVCodecParserContext *s);
104.
105. void (*av_init_packet)(AVPacket *pkt);
106. uint8_t* (*av_packet_new_side_data)(AVPacket *pkt, enum AVPacketSideDataType type, int size);
107.
108. int (*avcodec_h264_search_recovery_point)(AVCodecContext *avctx,
109.     const uint8_t *buf, int buf_size, int *recovery_frame_cnt);
110.
111. static const char_t *idctNames[], *errorRecognitions[], *errorConcealments[];
112. struct Tdia_size {
113.     int size;
114.     const char_t *descr;
115. };
116. static const Tdia_size dia_sizes[];
117.
118. //libswscale imports
119. SwsContext* (*sws_getCachedContext)(struct SwsContext *context, int srcW, int srcH, enum PixelFormat srcFormat,
120.     int dstW, int dstH, enum PixelFormat dstFormat, int flags,
121.     SwsFilter *srcFilter, SwsFilter *dstFilter, const double *param, SwsParams *ffdsow_params);
122.
123. void (*sws_freeContext)(SwsContext *c);
124. SwsFilter* (*sws_getDefaultFilter)(float lumaGBLur, float chromaGBLur,
125.     float lumaSharpen, float chromaSharpen,

```

```

126.         float chromaHShift, float chromaVShift,
127.         int verbose);
128. void (*sws_freeFilter)(SwsFilter *filter);
129. int sws_scale(struct SwsContext *context, const uint8_t* const srcSlice[], const stride_t srcStride[],
130.             int srcSliceY, int srcSliceH, uint8_t* const dst[], const stride_t dstStride[]);
131. SwsVector *(*sws_getConstVec)(double c, int length);
132. SwsVector *(*sws_getGaussianVec)(double variance, double quality);
133. void (*sws_normalizeVec)(SwsVector *a, double height);
134. void (*sws_freeVec)(SwsVector *a);
135. int (*sws_setColorspaceDetails)(struct SwsContext *c, const int inv_table[4],
136.                                int srcRange, const int table[4], int dstRange,
137.                                int brightness, int contrast, int saturation);
138. const int* (*sws_getCoefficients)(int colorspace);
139.
140. int (*GetCPUCount)(void);
141.
142. //libpostproc imports
143. void (*pp_postprocess)(const uint8_t * src[3], const stride_t srcStride[3],
144.                       uint8_t * dst[3], const stride_t dstStride[3],
145.                       int horizontalSize, int verticalSize,
146.                       const /*QP_STORE_T*/int8_t *QP_store, int QP_stride,
147.                       /*pp_mode*/void *mode, /*pp_context*/void *ppContext, int pict_type);
148.
149. /*pp_context*/
150. void (*pp_get_context)(int width, int height, int flags);
151. void (*pp_free_context)(/*pp_context*/void *ppContext);
152. void (*ff_simple_idct_mmx)(int16_t *block);
153.
154. // DXVA imports
155. int (*av_h264_decode_frame)(struct AVCodecContext* avctx, uint8_t *buf, int buf_size);
156. int (*av_vc1_decode_frame)(struct AVCodecContext* avctx, uint8_t *buf, int buf_size);
157.
158. // === H264 functions
159. int (*FFH264CheckCompatibility)(int nWidth, int nHeight, struct AVCodecContext* pAVCtx, BYTE* pBuffer, UINT nSize, int nPCIVendor,
160. int nPCIDevice, LARGE_INTEGER VideoDriverVersion);
161. int (*FFH264DecodeBuffer)
162. (struct AVCodecContext* pAVCtx, BYTE* pBuffer, UINT nSize, int* pFramePOC, int* pOutPOC, REFERENCE_TIME* pOutrtStart);
163. HRESULT(*FFH264BuildPicParams)(DXVA_PicParams_H264* pDXVAPicParams, DXVA_Qmatrix_H264* pDXVAScalingMatrix, int* nFieldType, int*
164. nSliceType, struct AVCodecContext* pAVCtx, int nPCIVendor);
165.
166. void (*FFH264SetCurrentPicture)(int nIndex, DXVA_PicParams_H264* pDXVAPicParams, struct AVCodecContext* pAVCtx);
167. void (*FFH264UpdateRefFramesList)(DXVA_PicParams_H264* pDXVAPicParams, struct AVCodecContext* pAVCtx);
168. BOOL (*FFH264IsRefFrameInUse)(int nFrameNum, struct AVCodecContext* pAVCtx);
169. void (*FFH264UpdateRefFrameSliceLong)(DXVA_PicParams_H264* pDXVAPicParams, DXVA_Slice_H264_Long* pSlice, struct AVCodecContext* p
170. AVCtx);
171. void (*FFH264SetDxvaSliceLong)(struct AVCodecContext* pAVCtx, void* pSliceLong);
172.
173. // === VC1 functions
174. HRESULT(*FFVC1UpdatePictureParam)(DXVA_PictureParameters* pPicParams, struct AVCodecContext* pAVCtx, int* nFieldType, int* nSlic
175. eType, BYTE* pBuffer, UINT nSize, UINT* nFrameSize, BOOL b_SecondField, BOOL* b_repeat_pict);
176. int (*FFIsSkipped)(struct AVCodecContext* pAVCtx);
177.
178. // === Common functions
179. char* (*GetFFMpegPictureType)(int nType);
180. unsigned long(*FFGetMBNumber)(struct AVCodecContext* pAVCtx);
181.
182. // yadif
183. void (*yadif_init)(YADIFContext *yadctx);
184. void (*yadif_uninit)(YADIFContext *yadctx);
185. void (*yadif_filter)(YADIFContext *yadctx, uint8_t *dst[3], stride_t dst_stride[3], int width, int height, int parity, int tff);
186.
187. // gradfun
188. int (*gradfunInit)(GradFunContext *ctx, const char *args, void *opaque);
189. void (*gradfunFilter)(GradFunContext *ctx, uint8_t *dst, uint8_t *src, int width, int height, int dst_linesize, int src_linesize
190. , int r);
191. };
192.
193. #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：

```

1.  /*
2.  *雷霄骅
3.  *leixiaohua1020@126.com
4.  *中国传媒大学/数字电视技术
5.  */
6.  #ifndef _TDLL_H_
7.  #define _TDLL_H_
8.
9.  #include "Tconfig.h"
10. //操作Dll的类
11. class Tdll

```

```

12. {
13. public:
14.     bool ok;
15.     Tdll(const char_t *dllName1, const Tconfig *config, bool explicitFullPath = false) {
16.         char_t name[MAX_PATH], ext[MAX_PATH];
17.         _splitpath_s(dllName1, NULL, 0, NULL, 0, name, countof(name), ext, countof(ext));
18.         if (config && !explicitFullPath) {
19.             char_t dllName2[MAX_PATH]; //installldir+filename+ext
20.             _makepath_s(dllName2, countof(dllName2), NULL, config->pth, name, ext);
21.             hdll = LoadLibrary(dllName2);
22.         } else {
23.             hdll = NULL;
24.         }
25.         if (!hdll) {
26.             hdll = LoadLibrary(dllName1);
27.             if (!hdll && !explicitFullPath) {
28.                 if (config) {
29.                     char_t dllName3[MAX_PATH]; //ffdsow.ax_path+filename+ext
30.                     _makepath_s(dllName3, countof(dllName3), NULL, config->epth, name, ext);
31.                     hdll = LoadLibrary(dllName3);
32.                 }
33.                 if (!hdll) {
34.                     char_t dllName0[MAX_PATH]; //only filename+ext - let Windows find it
35.                     _makepath_s(dllName0, countof(dllName0), NULL, NULL, name, ext);
36.                     hdll = LoadLibrary(dllName0);
37.                 }
38.             }
39.         }
40.         ok = (hdll != NULL);
41.     }
42.     ~Tdll() {
43.         if (hdll) {
44.             FreeLibrary(hdll);
45.         }
46.     }
47.     HMODULE hdll;
48.     //封装一下直接加载DLL的GetProcAddress
49.     template<class T> __forceinline void loadFunction(T &fnc, const char *name) {
50.         fnc = hdll ? (T)GetProcAddress(hdll, name) : NULL;
51.         ok &= (fnc != NULL);
52.     }
53.     template<class T> __forceinline void loadFunctionByIndex(T &fnc, uint16_t id) {
54.         uint32_t id32 = uint32_t(id);
55.         fnc = hdll ?
56.             (T) GetProcAddress(hdll, (LPCSTR)id32) :
57.             NULL;
58.         ok &= (fnc != NULL);
59.     }
60.     //检查DLL的状态是否正常
61.     static bool check(const char_t *dllName1, const Tconfig *config) {
62.         char_t name[MAX_PATH], ext[MAX_PATH];
63.         _splitpath_s(dllName1, NULL, 0, NULL, 0, name, countof(name), ext, countof(ext));
64.         if (config) {
65.             char_t dllName2[MAX_PATH]; //installldir+filename+ext
66.             _makepath_s(dllName2, countof(dllName2), NULL, config->pth, name, ext);
67.             if (fileexists(dllName2)) {
68.                 return true;
69.             }
70.         }
71.         if (fileexists(dllName1)) {
72.             return true;
73.         }
74.         if (config) {
75.             char_t dllName3[MAX_PATH]; //ffdsow.ax_path+filename+ext
76.             _makepath_s(dllName3, MAX_PATH, NULL, config->epth, name, ext);
77.             if (fileexists(dllName3)) {
78.                 return true;
79.             }
80.         }
81.         char_t dllName0[MAX_PATH]; //only filename+ext - let Windows find it
82.         _makepath_s(dllName0, countof(dllName0), NULL, NULL, name, ext);
83.         char_t dir0[MAX_PATH], *dir0flnm;
84.         if (SearchPath(NULL, dllName0, NULL, MAX_PATH, dir0, &dir0flnm)) {
85.             return true;
86.         }
87.         return false;
88.     }
89. };
90.
91. #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：

```

1. //===== Tlibavcodec =====
2. //FFMPEG封装类的构造函数
3. Tlibavcodec::Tlibavcodec(const Tconfig *config): refcount(0)
4. {
5.     //加载FFMPEG的Dll
6.     dll = new Tdll(_L("ffmpeg.dll"), config);
7.     //加载各个函数
8.     dll->loadFunction(avcodec_register_all, "avcodec_register_all");
9.     dll->loadFunction(avcodec_find_decoder, "avcodec_find_decoder");
10.    dll->loadFunction(avcodec_open0, "avcodec_open2");
11.    dll->loadFunction(avcodec_alloc_context0, "avcodec_alloc_context3");
12.    dll->loadFunction(avcodec_alloc_frame, "avcodec_alloc_frame");
13.    dll->loadFunction(avcodec_decode_video2, "avcodec_decode_video2");
14.    dll->loadFunction(avcodec_flush_buffers, "avcodec_flush_buffers");
15.    dll->loadFunction(avcodec_close0, "avcodec_close");
16.    dll->loadFunction(av_log_set_callback, "av_log_set_callback");
17.    dll->loadFunction(av_log_get_callback, "av_log_get_callback");
18.    dll->loadFunction(av_log_get_level, "av_log_get_level");
19.    dll->loadFunction(av_log_set_level, "av_log_set_level");
20.    dll->loadFunction(av_set_cpu_flags_mask, "av_set_cpu_flags_mask");
21.    dll->loadFunction(av_mallocz, "av_mallocz");
22.    dll->loadFunction(av_free, "av_free");
23.    dll->loadFunction(avcodec_default_get_buffer, "avcodec_default_get_buffer");
24.    dll->loadFunction(avcodec_default_release_buffer, "avcodec_default_release_buffer");
25.    dll->loadFunction(avcodec_default_reget_buffer, "avcodec_default_reget_buffer");
26.    dll->loadFunction(avcodec_get_current_idct, "avcodec_get_current_idct");
27.    dll->loadFunction(avcodec_get_encoder_info, "avcodec_get_encoder_info");
28.    dll->loadFunction(av_init_packet, "av_init_packet");
29.    dll->loadFunction(av_packet_new_side_data, "av_packet_new_side_data");
30.    dll->loadFunction(avcodec_h264_search_recovery_point, "avcodec_h264_search_recovery_point");
31.
32.    dll->loadFunction(avcodec_decode_audio3, "avcodec_decode_audio3");
33.
34.    dll->loadFunction(avcodec_find_encoder, "avcodec_find_encoder");
35.    dll->loadFunction(avcodec_encode_video, "avcodec_encode_video");
36.    dll->loadFunction(avcodec_encode_audio, "avcodec_encode_audio");
37.
38.    dll->loadFunction(av_parser_init, "av_parser_init");
39.    dll->loadFunction(av_parser_parse2, "av_parser_parse2");
40.    dll->loadFunction(av_parser_close, "av_parser_close");
41.
42.    //libswscale methods
43.    dll->loadFunction(sws_getCachedContext, "sws_getCachedContext");
44.    dll->loadFunction(sws_freeContext, "sws_freeContext");
45.    dll->loadFunction(sws_getDefaultFilter, "sws_getDefaultFilter");
46.    dll->loadFunction(sws_freeFilter, "sws_freeFilter");
47.    dll->loadFunction(libswscale_sws_scale, "sws_scale");
48.
49.    dll->loadFunction(GetCPUCount, "GetCPUCount");
50.    dll->loadFunction(sws_getConstVec, "sws_getConstVec");
51.    dll->loadFunction(sws_getGaussianVec, "sws_getGaussianVec");
52.    dll->loadFunction(sws_normalizeVec, "sws_normalizeVec");
53.    dll->loadFunction(sws_freeVec, "sws_freeVec");
54.    dll->loadFunction(sws_setColorspaceDetails, "sws_setColorspaceDetails");
55.    dll->loadFunction(sws_getCoefficients, "sws_getCoefficients");
56.
57.    //libpostproc methods
58.    dll->loadFunction(pp_postprocess, "pp_postprocess");
59.    dll->loadFunction(pp_get_context, "pp_get_context");
60.    dll->loadFunction(pp_free_context, "pp_free_context");
61.    dll->loadFunction(ff_simple_idct_mmx, "ff_simple_idct_mmx");
62.
63.    //DXVA methods
64.    dll->loadFunction(av_h264_decode_frame, "av_h264_decode_frame");
65.    dll->loadFunction(av_vcl_decode_frame, "av_vcl_decode_frame");
66.
67.    dll->loadFunction(FFH264CheckCompatibility, "FFH264CheckCompatibility");
68.    dll->loadFunction(FFH264DecodeBuffer, "FFH264DecodeBuffer");
69.    dll->loadFunction(FFH264BuildPicParams, "FFH264BuildPicParams");
70.    dll->loadFunction(FFH264SetCurrentPicture, "FFH264SetCurrentPicture");
71.    dll->loadFunction(FFH264UpdateRefFramesList, "FFH264UpdateRefFramesList");
72.    dll->loadFunction(FFH264IsRefFrameInUse, "FFH264IsRefFrameInUse");
73.    dll->loadFunction(FF264UpdateRefFrameSliceLong, "FF264UpdateRefFrameSliceLong");
74.    dll->loadFunction(FFH264SetDxvaSliceLong, "FFH264SetDxvaSliceLong");
75.
76.    dll->loadFunction(FFVC1UpdatePictureParam, "FFVC1UpdatePictureParam");
77.    dll->loadFunction(FFIsSkipped, "FFIsSkipped");
78.
79.    dll->loadFunction(GetFFMpegPictureType, "GetFFMpegPictureType");
80.    dll->loadFunction(FFGetMBNumber, "FFGetMBNumber");
81.
82.    //yadif methods
83.    dll->loadFunction(yadif_init, "yadif_init");

```

```

84.     dll->loadFunction(yadif_uninit, "yadif_uninit");
85.     dll->loadFunction(yadif_filter, "yadif_filter");
86.
87.     //gradfun
88.     dll->loadFunction(gradfunInit, "gradfunInit");
89.     dll->loadFunction(gradfunFilter, "gradfunFilter");
90.
91.     ok = dll->ok;
92.     //加载完毕后, 进行注册
93.     if (ok) {
94.         avcodec_register_all();
95.         av_log_set_callback(avlog);
96.     }
97. }

```

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:

```

1. Tlibavcodec::~Tlibavcodec()
2. {
3.     delete dll;
4. }

```

检查Dll的函数也比较简单:

```

1. bool Tlibavcodec::check(const Tconfig *config)
2. {
3.     return Tdll::check(_l("ffmpeg.dll"), config);
4. }

```

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:

```

1. bool Tlibavcodec::getVersion(const Tconfig *config, ffstring &vers, ffstring &license)
2. {
3.     Tdll *dl = new Tdll(_l("ffmpeg.dll"), config);
4.
5.     void (*av_getVersion)(char **version, char **build, char **datetime, const char* *license);
6.     dl->loadFunction(av_getVersion, "getVersion");
7.     bool res;
8.     if (av_getVersion) {
9.         res = true;
10.         char *version, *build, *datetime;
11.         const char *lic;
12.         av_getVersion(&version, &build, &datetime, &lic);
13.         vers = (const char_t*)text<char_t>(version) + ffstring(_l(" ")) + (const char_t*)text<char_t>(datetime) + _l(" ");
14.         license = text<char_t>(lic);
15.     } else {
16.         res = false;
17.         vers.clear();
18.         license.clear();
19.     }
20.     delete dl;
21.     return res;
22. }

```

个人分类：[ffdsnow](#)

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:liushide@163.com