

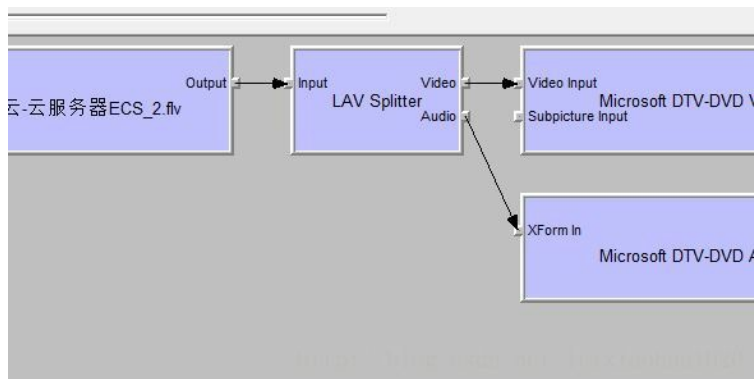
原 LAV Filter 源代码分析 2：LAV Splitter

2013年10月14日 21:06:30 阅读数：9445

LAV Filter 中最著名的就是 LAV Splitter,支持Matroska /WebM,MPEG-TS/PS,MP4/MOV,FLV,OGM / OGG,AVI等其他格式,广泛存在于各种视频播放器（暴风影音这类的）之中。

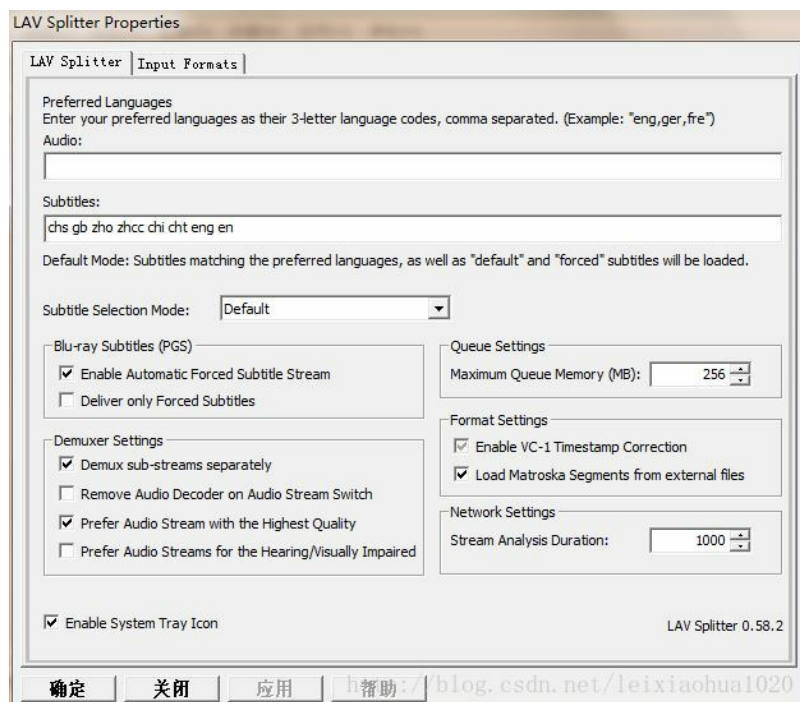
本文分析一下它的源代码。在分析之前,先看看它是什么样的。

使用GraphEdit随便打开一个视频文件,就可以看见LAV Filter：

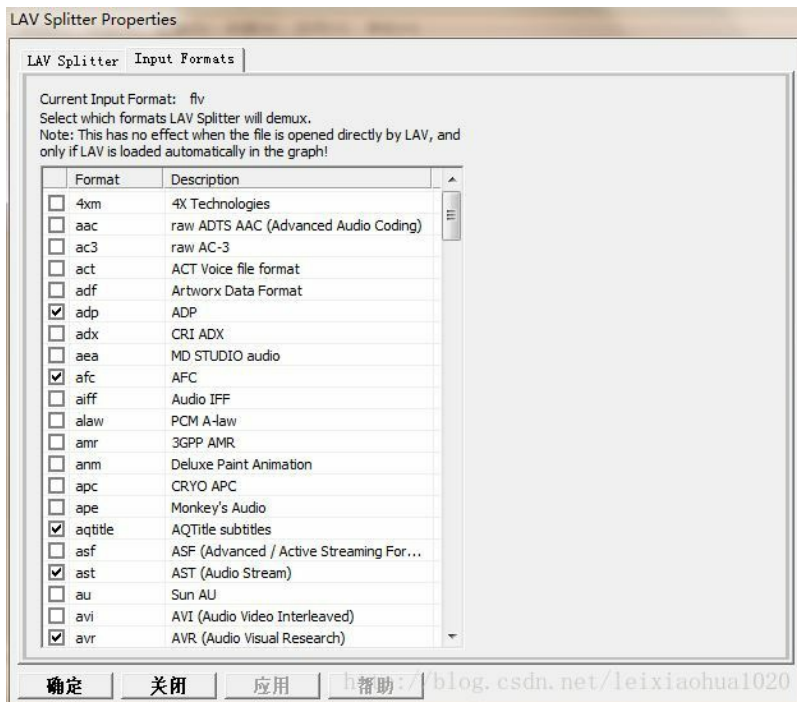


可以右键点击这个Filter看一下它的属性页面,如图所示：

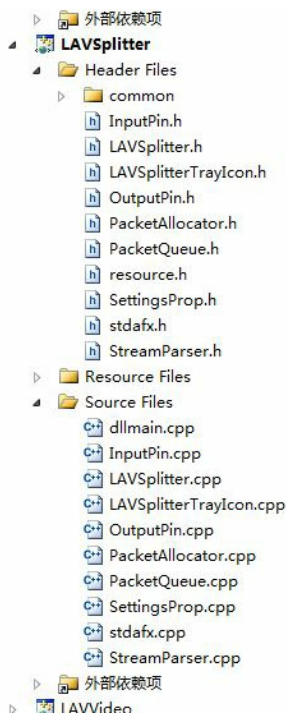
属性设置页面：



支持输入格式：



下面我们在 VC 2010 中看一下它的源代码：



从何看起呢？就先从directshow的注册函数看起吧，位于dllmain.cpp之中。部分代码的含义已经用注释标注上了。从代码可以看出，和普通的DirectShow Filter没什么区别。

dllmain.cpp

```
[cpp]
1.  /*
2.  *      Copyright (C) 2010-2013 Hendrik Leppkes
3.  *      http://www.1f0.de
4.  *
5.  *      This program is free software; you can redistribute it and/or modify
6.  *      it under the terms of the GNU General Public License as published by
7.  *      the Free Software Foundation; either version 2 of the License, or
8.  *      (at your option) any later version.
9.  *
10. *      This program is distributed in the hope that it will be useful,
11. *      but WITHOUT ANY WARRANTY; without even the implied warranty of
12. *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13. *      GNU General Public License for more details.
14. *
15. *      You should have received a copy of the GNU General Public License along
16. *      with this program; if not, write to the Free Software Foundation, Inc.,
17. *      51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
```

```

18.  */
19.
20.  // Based on the SampleParser Template by GDCL
21.  // -----
22.  // Copyright (c) GDCL 2004. All Rights Reserved.
23.  // You are free to re-use this as the basis for your own filter development,
24.  // provided you retain this copyright notice in the source.
25.  // http://www.gdcl.co.uk
26.  // -----
27.  //各种定义。。。
28.  #include "stdafx.h"
29.
30.  // Initialize the GUIDs
31.  #include <InitGuid.h>
32.
33.  #include <qnetwork.h>
34.  #include "LAVSplitter.h"
35.  #include "moreuuids.h"
36.
37.  #include "registry.h"
38.  #include "IGraphRebuildDelegate.h"
39.
40.  // The GUID we use to register the splitter media types
41.  DEFINE_GUID(MEDIATYPE_LAVSplitter,
42.      0x9c53931c, 0x7d5a, 0x4a75, 0xb2, 0x6f, 0x4e, 0x51, 0x65, 0x4d, 0xb2, 0xc0);
43.
44.  // --- COM factory table and registration code -----
45.  //注册时候的信息
46.  const AMOVIESETUP_MEDIATYPE
47.      sudMediaTypes[] = {
48.      { &MEDIATYPE_Stream, &MEDIASUBTYPE_NULL },
49.  };
50.  //注册时候的信息 (PIN)
51.  const AMOVIESETUP_PIN sudOutputPins[] =
52.  {
53.      {
54.          L"Output",          // pin name
55.          FALSE,              // is rendered?
56.          TRUE,               // is output?
57.          FALSE,              // zero instances allowed?
58.          TRUE,               // many instances allowed?
59.          &CLSID_NULL,        // connects to filter (for bridge pins)
60.          NULL,               // connects to pin (for bridge pins)
61.          0,                  // count of registered media types
62.          NULL                // list of registered media types
63.      },
64.      {
65.          L"Input",           // pin name
66.          FALSE,              // is rendered?
67.          FALSE,              // is output?
68.          FALSE,              // zero instances allowed?
69.          FALSE,              // many instances allowed?
70.          &CLSID_NULL,        // connects to filter (for bridge pins)
71.          NULL,               // connects to pin (for bridge pins)
72.          1,                  // count of registered media types
73.          &sudMediaTypes[0]   // list of registered media types
74.      }
75.  };
76.  //注册时候的信息 (名称等)
77.  //CLAVSplitter
78.  const AMOVIESETUP_FILTER sudFilterReg =
79.  {
80.      &_uuidof(CLAVSplitter), // filter clsid
81.      L"LAV Splitter",        // filter name
82.      MERIT_PREFERRED + 4,    // merit
83.      2,                       // count of registered pins
84.      sudOutputPins,          // list of pins to register
85.      CLSID_LegacyAmFilterCategory
86.  };
87.  //注册时候的信息 (名称等)
88.  //CLAVSplitterSource
89.  const AMOVIESETUP_FILTER sudFilterRegSource =
90.  {
91.      &_uuidof(CLAVSplitterSource), // filter clsid
92.      L"LAV Splitter Source",        // filter name
93.      MERIT_PREFERRED + 4,           // merit
94.      1,                             // count of registered pins
95.      sudOutputPins,                 // list of pins to register
96.      CLSID_LegacyAmFilterCategory
97.  };
98.
99.  // --- COM factory table and registration code -----
100.
101.  // DirectShow base class COM factory requires this table,
102.  // declaring all the COM objects in this DLL
103.  // 注意g_Templates名称是固定的
104.  CFactoryTemplate g_Templates[] = {
105.      // one entry for each CoCreate-able object
106.      {
107.          sudFilterReg.strName,
108.          sudFilterReg.clsID,
109.          CoCreateInstance(&CLAVSplitter,

```

```

109.         CreateInstance<CLAVSplitter>,
110.         CLAVSplitter::StaticInit,
111.         &sudFilterReg
112.     },
113.     {
114.         sudFilterRegSource.strName,
115.         sudFilterRegSource.clsID,
116.         CreateInstance<CLAVSplitterSource>,
117.         NULL,
118.         &sudFilterRegSource
119.     },
120.     // This entry is for the property page.
121.     // 属性页
122.     {
123.         L"LAV Splitter Properties",
124.         &CLSID_LAVSplitterSettingsProp,
125.         CreateInstance<CLAVSplitterSettingsProp>,
126.         NULL, NULL
127.     },
128.     {
129.         L"LAV Splitter Input Formats",
130.         &CLSID_LAVSplitterFormatsProp,
131.         CreateInstance<CLAVSplitterFormatsProp>,
132.         NULL, NULL
133.     }
134. };
135. int g_cTemplates = sizeof(g_Templates) / sizeof(g_Templates[0]);
136.
137. // self-registration entrypoint
138. STDAPI DllRegisterServer()
139. {
140.     std::list<LPCWSTR> chkbytes;
141.
142.     // BluRay
143.     chkbytes.clear();
144.     chkbytes.push_back(L"0,4,,494E4458"); // INDX (index.bdmv)
145.     chkbytes.push_back(L"0,4,,4D4F424A"); // MOBJ (MovieObject.bdmv)
146.     chkbytes.push_back(L"0,4,,4D504C53"); // MPLS
147.     RegisterSourceFilter(__uuidof(CLAVSplitterSource),
148.         MEDIASUBTYPE_LAVBluRay, chkbytes, NULL);
149.
150.     // base classes will handle registration using the factory template table
151.     return AMovieDllRegisterServer2(true);
152. }
153.
154. STDAPI DllUnregisterServer()
155. {
156.     UnRegisterSourceFilter(MEDIASUBTYPE_LAVBluRay);
157.
158.     // base classes will handle de-registration using the factory template table
159.     return AMovieDllRegisterServer2(false);
160. }
161.
162. // if we declare the correct C runtime entrypoint and then forward it to the DShow base
163. // classes we will be sure that both the C/C++ runtimes and the base classes are initialized
164. // correctly
165. extern "C" BOOL WINAPI DllEntryPoint(HINSTANCE, ULONG, LPVOID);
166. BOOL WINAPI DllMain(HANDLE hDllHandle, DWORD dwReason, LPVOID lpReserved)
167. {
168.     return DllEntryPoint(reinterpret_cast<HINSTANCE>(hDllHandle), dwReason, lpReserved);
169. }
170.
171. void CALLBACK OpenConfiguration(HWND hwnd, HINSTANCE hinst, LPSTR lpszCmdLine, int nCmdShow)
172. {
173.     HRESULT hr = S_OK;
174.     CUnknown *pInstance = CreateInstance<CLAVSplitter>(NULL, &hr);
175.     IBaseFilter *pFilter = NULL;
176.     pInstance->NonDelegatingQueryInterface(IID_IBaseFilter, (void **)&pFilter);
177.     if (pFilter) {
178.         pFilter->AddRef();
179.         CBaseDSPropPage::ShowPropPageDialog(pFilter);
180.     }
181.     delete pInstance;
182. }

```

接下来就要进入正题了，看一看核心的分离器（解封装器）的类CLAVSplitter的定义文件LAVSplitter.h。乍一看这个类确实了得，居然继承了那么多的父类，实在是碉堡了。先不管那么多，看看里面都有什么函数吧。主要的函数上面都加了注释。注意还有一个类CLAVSplitterSource继承了CLAVSplitter。

LAVSplitter.h

```

1.  /*
2.  *   Copyright (C) 2010-2013 Hendrik Leppkes
3.  *   http://www.1f0.de
4.  *
5.  *   This program is free software; you can redistribute it and/or modify
6.  *   it under the terms of the GNU General Public License as published by
7.  *   the Free Software Foundation; either version 2 of the License, or
8.  *   (at your option) any later version.

```

```

9.  *
10. * This program is distributed in the hope that it will be useful,
11. * but WITHOUT ANY WARRANTY; without even the implied warranty of
12. * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13. * GNU General Public License for more details.
14. *
15. * You should have received a copy of the GNU General Public License along
16. * with this program; if not, write to the Free Software Foundation, Inc.,
17. * 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
18. *
19. * Initial design and concept by Gabest and the MPC-HC Team, copyright under GPLv2
20. * Contributions by Ti-BEN from the XBMC DSPlayer Project, also under GPLv2
21. */
22.
23. #pragma once
24.
25. #include <string>
26. #include <list>
27. #include <set>
28. #include <vector>
29. #include <map>
30. #include "PacketQueue.h"
31.
32. #include "BaseDemuxer.h"
33.
34. #include "LAVSplitterSettingsInternal.h"
35. #include "SettingsProp.h"
36. #include "IBufferInfo.h"
37.
38. #include "ISpecifyPropertyPages2.h"
39.
40. #include "LAVSplitterTrayIcon.h"
41.
42. #define LAVF_REGISTRY_KEY L"Software\\LAV\\Splitter"
43. #define LAVF_REGISTRY_KEY_FORMATS LAVF_REGISTRY_KEY L"\\Formats"
44. #define LAVF_LOG_FILE L"LAVSplitter.txt"
45.
46. #define MAX_PTS_SHIFT 50000000i64
47.
48. class CLAVOutputPin;
49. class CLAVInputPin;
50.
51. #ifdef _MSC_VER
52. #pragma warning(disable: 4355)
53. #endif
54. //核心解复用 (分离器)
55. //暴露的接口在ILAVFSettings中
56. [uuid("171252A0-8820-4AFE-9DF8-5C92B2D66B04")]
57. class LAVSplitter
58. : public CBaseFilter
59. , public CCritSec
60. , protected CAMThread
61. , public IFileSourceFilter
62. , public IMediaSeeking
63. , public IAMStreamSelect
64. , public IAMOpenProgress
65. , public ILAVFSettingsInternal
66. , public ISpecifyPropertyPages2
67. , public IObjectWithSite
68. , public IBufferInfo
69. {
70. public:
71. LAVSplitter(LPUNKNOWN pUnk, HRESULT* phr);
72. virtual ~LAVSplitter();
73.
74. static void CALLBACK StaticInit(BOOL bLoading, const CLSID *clsid);
75.
76. // IUnknown
77. //
78. DECLARE_IUNKNOWN;
79. //暴露接口, 使外部程序可以QueryInterface, 关键!
80. //翻译 ("没有代表的方式查询接口")
81. STDMETHODIMP NonDelegatingQueryInterface(REFIID riid, void** ppv);
82.
83. // CBaseFilter methods
84. //输入是一个, 输出就不一定了!
85. int GetPinCount();
86. CBasePin *GetPin(int n);
87. STDMETHODIMP GetClassID(CLSID* pClsID);
88.
89. STDMETHODIMP Stop();
90. STDMETHODIMP Pause();
91. STDMETHODIMP Run(REFERENCE_TIME tStart);
92.
93. STDMETHODIMP JoinFilterGraph(IFilterGraph * pGraph, LPCWSTR pName);
94.
95. // IFileSourceFilter
96. // 源Filter的接口方法
97. STDMETHODIMP Load(LPCOLESTR pszFileName, const AM_MEDIA_TYPE * pmt);
98. STDMETHODIMP GetCurFile(LPOLESTR *ppszFileName, AM_MEDIA_TYPE *pmt);
99.

```

```

100. // IMediaSeeking
101. STDMETHODCALLTYPE GetCapabilities(DWORD* pCapabilities);
102. STDMETHODCALLTYPE CheckCapabilities(DWORD* pCapabilities);
103. STDMETHODCALLTYPE IsFormatSupported(const GUID* pFormat);
104. STDMETHODCALLTYPE QueryPreferredFormat(GUID* pFormat);
105. STDMETHODCALLTYPE GetTimeFormat(GUID* pFormat);
106. STDMETHODCALLTYPE IsUsingTimeFormat(const GUID* pFormat);
107. STDMETHODCALLTYPE SetTimeFormat(const GUID* pFormat);
108. STDMETHODCALLTYPE GetDuration(LONGLONG* pDuration);
109. STDMETHODCALLTYPE GetStopPosition(LONGLONG* pStop);
110. STDMETHODCALLTYPE GetCurrentPosition(LONGLONG* pCurrent);
111. STDMETHODCALLTYPE ConvertTimeFormat(LONGLONG* pTarget, const GUID* pTargetFormat, LONGLONG Source, const GUID* pSourceFormat);
112. STDMETHODCALLTYPE SetPositions(LONGLONG* pCurrent, DWORD dwCurrentFlags, LONGLONG* pStop, DWORD dwStopFlags);
113. STDMETHODCALLTYPE GetPositions(LONGLONG* pCurrent, LONGLONG* pStop);
114. STDMETHODCALLTYPE GetAvailable(LONGLONG* pEarliest, LONGLONG* pLatest);
115. STDMETHODCALLTYPE SetRate(double dRate);
116. STDMETHODCALLTYPE GetRate(double* pdRate);
117. STDMETHODCALLTYPE GetPreroll(LONGLONG* pllPreroll);
118.
119. // IAMStreamSelect
120. STDMETHODCALLTYPE Count(DWORD *pcStreams);
121. STDMETHODCALLTYPE Enable(long lIndex, DWORD dwFlags);
122. STDMETHODCALLTYPE Info(long lIndex, AM_MEDIA_TYPE **ppmt, DWORD *pdwFlags, LCID *plcid, DWORD *pdwGroup, WCHAR **ppszName, IUnknown **ppObject, IUnknown **ppUnk);
123.
124. // IAMOpenProgress
125. STDMETHODCALLTYPE QueryProgress(LONGLONG *pllTotal, LONGLONG *pllCurrent);
126. STDMETHODCALLTYPE AbortOperation();
127.
128. // ISpecifyPropertyPages2
129. STDMETHODCALLTYPE GetPages(CAUUID *pPages);
130. STDMETHODCALLTYPE CreatePage(const GUID& guid, IPropertyPage** ppPage);
131.
132. // IObjectWithSite
133. STDMETHODCALLTYPE SetSite(IUnknown *pUnkSite);
134. STDMETHODCALLTYPE GetSite(REFIID riid, void **ppvSite);
135.
136. // IBufferInfo
137. STDMETHODCALLTYPE_(int) GetCount();
138. STDMETHODCALLTYPE GetStatus(int i, int& samples, int& size);
139. STDMETHODCALLTYPE_(DWORD) GetPriority();
140.
141. // ILAVFSettings
142. STDMETHODCALLTYPE SetRuntimeConfig(BOOL bRuntimeConfig);
143. STDMETHODCALLTYPE GetPreferredLanguages(LPWSTR *ppLanguages);
144. STDMETHODCALLTYPE SetPreferredLanguages(LPCWSTR pLanguages);
145. STDMETHODCALLTYPE GetPreferredSubtitleLanguages(LPWSTR *ppLanguages);
146. STDMETHODCALLTYPE SetPreferredSubtitleLanguages(LPCWSTR pLanguages);
147. STDMETHODCALLTYPE_(LAVSubtitleMode) GetSubtitleMode();
148. STDMETHODCALLTYPE SetSubtitleMode(LAVSubtitleMode mode);
149. STDMETHODCALLTYPE_(BOOL) GetSubtitleMatchingLanguage();
150. STDMETHODCALLTYPE SetSubtitleMatchingLanguage(BOOL dwMode);
151. STDMETHODCALLTYPE_(BOOL) GetPGSForcedStream();
152. STDMETHODCALLTYPE SetPGSForcedStream(BOOL bFlag);
153. STDMETHODCALLTYPE_(BOOL) GetPGSOnlyForced();
154. STDMETHODCALLTYPE SetPGSOnlyForced(BOOL bForced);
155. STDMETHODCALLTYPE_(int) GetVCLTimestampMode();
156. STDMETHODCALLTYPE SetVCLTimestampMode(int iMode);
157. STDMETHODCALLTYPE SetSubstreamsEnabled(BOOL bSubStreams);
158. STDMETHODCALLTYPE_(BOOL) GetSubstreamsEnabled();
159. STDMETHODCALLTYPE SetVideoParsingEnabled(BOOL bEnabled);
160. STDMETHODCALLTYPE_(BOOL) GetVideoParsingEnabled();
161. STDMETHODCALLTYPE SetFixBrokenHDPVR(BOOL bEnabled);
162. STDMETHODCALLTYPE_(BOOL) GetFixBrokenHDPVR();
163. STDMETHODCALLTYPE_(HRESULT) SetFormatEnabled(LPCSTR strFormat, BOOL bEnabled);
164. STDMETHODCALLTYPE_(BOOL) IsFormatEnabled(LPCSTR strFormat);
165. STDMETHODCALLTYPE SetStreamSwitchRemoveAudio(BOOL bEnabled);
166. STDMETHODCALLTYPE_(BOOL) GetStreamSwitchRemoveAudio();
167. STDMETHODCALLTYPE GetAdvancedSubtitleConfig(LPWSTR *ppAdvancedConfig);
168. STDMETHODCALLTYPE SetAdvancedSubtitleConfig(LPCWSTR pAdvancedConfig);
169. STDMETHODCALLTYPE SetUseAudioForHearingVisuallyImpaired(BOOL bEnabled);
170. STDMETHODCALLTYPE_(BOOL) GetUseAudioForHearingVisuallyImpaired();
171. STDMETHODCALLTYPE SetMaxQueueMemSize(DWORD dwMaxSize);
172. STDMETHODCALLTYPE_(DWORD) GetMaxQueueMemSize();
173. STDMETHODCALLTYPE SetTrayIcon(BOOL bEnabled);
174. STDMETHODCALLTYPE_(BOOL) GetTrayIcon();
175. STDMETHODCALLTYPE SetPreferHighQualityAudioStreams(BOOL bEnabled);
176. STDMETHODCALLTYPE_(BOOL) GetPreferHighQualityAudioStreams();
177. STDMETHODCALLTYPE SetLoadMatroskaExternalSegments(BOOL bEnabled);
178. STDMETHODCALLTYPE_(BOOL) GetLoadMatroskaExternalSegments();
179. STDMETHODCALLTYPE GetFormats(LPSTR** formats, UINT* nFormats);
180. STDMETHODCALLTYPE SetNetworkStreamAnalysisDuration(DWORD dwDuration);
181. STDMETHODCALLTYPE_(DWORD) GetNetworkStreamAnalysisDuration();
182.
183. // ILAVSplitterSettingsInternal
184. STDMETHODCALLTYPE_(LPCSTR) GetInputFormat() { if (m_pDemuxer) return m_pDemuxer->GetContainerFormat(); return NULL; }
185. STDMETHODCALLTYPE_(std::set<FormatInfo>&) GetInputFormats();
186. STDMETHODCALLTYPE_(BOOL) IsVCLCorrectionRequired();
187. STDMETHODCALLTYPE_(CMediaType *) GetOutputMediaType(int stream);
188. STDMETHODCALLTYPE_(IFilterGraph *) GetFilterGraph() { if (m_pGraph) { m_pGraph->AddRef(); return m_pGraph; } return NULL; }
189.

```

```

190.     STDMETHODIMP_(DWORD) GetStreamFlags(DWORD dwStream) { if (m_pDemuxer) return m_pDemuxer->GetStreamFlags(dwStream); return 0; }
191.     STDMETHODIMP_(int) GetPixelFormat(DWORD dwStream) { if (m_pDemuxer) return m_pDemuxer->GetPixelFormat(dwStream); return AV_PIX_FMT
    _NONE; }
192.     STDMETHODIMP_(int) GetHasBFrames(DWORD dwStream){ if (m_pDemuxer) return m_pDemuxer->GetHasBFrames(dwStream); return -1; }
193.
194.     // Settings helper
195.     std::list<std::string> GetPreferredAudioLanguageList();
196.     std::list<CSubtitleSelector> GetSubtitleSelectors();
197.
198.     bool IsAnyPinDrying();
199.     void SetFakeASFReader(BOOL bFlag) { m_bFakeASFReader = bFlag; }
200. protected:
201.     // CAMThread
202.     enum {CMD_EXIT, CMD_SEEK};
203.     DWORD ThreadProc();
204.
205.     HRESULT DemuxSeek(REFERENCE_TIME rtStart);
206.     HRESULT DemuxNextPacket();
207.     HRESULT DeliverPacket(Packet *pPacket);
208.
209.     void DeliverBeginFlush();
210.     void DeliverEndFlush();
211.
212.     STDMETHODIMP Close();
213.     STDMETHODIMP DeleteOutputs();
214.     //初始化解复用器
215.     STDMETHODIMP InitDemuxer();
216.
217.     friend class CLAVOutputPin;
218.     STDMETHODIMP SetPositionsInternal(void *caller, LONGLONG* pCurrent, DWORD dwCurrentFlags, LONGLONG* pStop, DWORD dwStopFlags);
219.
220. public:
221.     CLAVOutputPin *GetOutputPin(DWORD streamId, BOOL bActiveOnly = FALSE);
222.     STDMETHODIMP RenameOutputPin(DWORD TrackNumSrc, DWORD TrackNumDst, std::vector<CMediaType> pmts);
223.     STDMETHODIMP UpdateForcedSubtitleMediaType();
224.
225.     STDMETHODIMP CompleteInputConnection();
226.     STDMETHODIMP BreakInputConnection();
227.
228. protected:
229.     //相关的参数设置
230.     STDMETHODIMP LoadDefaults();
231.     STDMETHODIMP ReadSettings(HKEY rootKey);
232.     STDMETHODIMP LoadSettings();
233.     STDMETHODIMP SaveSettings();
234.     //创建图标
235.     STDMETHODIMP CreateTrayIcon();
236.
237. protected:
238.     CLAVInputPin *m_pInput;
239.
240. private:
241.     CCritSec m_csPins;
242.     //用vector存储输出PIN（解复用的时候是不确定的）
243.     std::vector<CLAVOutputPin *> m_pPins;
244.     //活动的
245.     std::vector<CLAVOutputPin *> m_pActivePins;
246.     //不用的
247.     std::vector<CLAVOutputPin *> m_pRetiredPins;
248.     std::set<DWORD> m_bDiscontinuitySent;
249.
250.     std::wstring m_fileName;
251.     std::wstring m_processName;
252.     //有很多纯虚函数的基本解复用类
253.     //注意：绝大部分信息都是从这获得的
254.     //这里的信息是由其派生类从FFMPEG中获取到的
255.     CBaseDemuxer *m_pDemuxer;
256.
257.     BOOL m_bPlaybackStarted;
258.     BOOL m_bFakeASFReader;
259.
260.     // Times
261.     REFERENCE_TIME m_rtStart, m_rtStop, m_rtCurrent, m_rtNewStart, m_rtNewStop;
262.     REFERENCE_TIME m_rtOffset;
263.     double m_dRate;
264.     BOOL m_bStopValid;
265.
266.     // Seeking
267.     REFERENCE_TIME m_rtLastStart, m_rtLastStop;
268.     std::set<void *> m_LastSeekers;
269.
270.     // flushing
271.     bool m_fflushing;
272.     CAMEvent m_eEndFlush;
273.
274.     std::set<FormatInfo> m_InputFormats;
275.
276.     // Settings
277.     //设置
278.     struct Settings {
279.         BOOL TrayIcon;
280.

```

```

280.     std::wstring prefAudioLangs;
281.     std::wstring prefSubLangs;
282.     std::wstring subtitleAdvanced;
283.     LAVSubtitleMode subtitleMode;
284.     BOOL PGSForcedStream;
285.     BOOL PGSOnlyForced;
286.     int vc1Mode;
287.     BOOL substreams;
288.
289.     BOOL MatroskaExternalSegments;
290.
291.     BOOL StreamSwitchRemoveAudio;
292.     BOOL ImpairedAudio;
293.     BOOL PreferHighQualityAudio;
294.     DWORD QueueMaxSize;
295.     DWORD NetworkAnalysisDuration;
296.
297.     std::map<std::string, BOOL> formats;
298. } m_settings;
299.
300. BOOL m_bRuntimeConfig;
301.
302. IUnknown *m_pSite;
303.
304. CBaseTrayIcon *m_pTrayIcon;
305. };
306.
307. [uuid("B98D13E7-55DB-4385-A33D-09FD1BA26338")]
308. class CLAVSplitterSource : public CLAVSplitter
309. {
310. public:
311.     // construct only via class factory
312.     CLAVSplitterSource(LPUNKNOWN pUnk, HRESULT* phr);
313.     virtual ~CLAVSplitterSource();
314.
315.     // IUnknown
316.     DECLARE_IUNKNOWN;
317.     //暴露接口，使外部程序可以QueryInterface，关键！
318.     //翻译（“没有代表的方式查询接口”）
319.     STDMETHODIMP NonDelegatingQueryInterface(REFIID riid, void** ppv);
320. };

```

先来看一下查询接口的函数NonDelegatingQueryInterface()吧

```

1. //暴露接口，使外部程序可以QueryInterface，关键！
2. STDMETHODIMP CLAVSplitter::NonDelegatingQueryInterface(REFIID riid, void** ppv)
3. {
4.     CheckPointer(ppv, E_POINTER);
5.
6.     *ppv = NULL;
7.
8.     if (m_pDemuxer && (riid == __uuidof(IKeyFrameInfo) || riid == __uuidof(ITrackInfo) || riid == IID_IAMExtendedSeeking || riid == IID_MMediaContent)) {
9.         return m_pDemuxer->QueryInterface(riid, ppv);
10.    }
11.    //写法好特别啊，意思是一样的
12.    return
13.        QI(IMediaSeeking)
14.        QI(IAMStreamSelect)
15.        QI(ISpecifyPropertyPages)
16.        QI(ISpecifyPropertyPages2)
17.        QI2(ILAVFSettings)
18.        QI2(ILAVFSettingsInternal)
19.        QI(IObjectWithSite)
20.        QI(IBufferInfo)
21.        __super::NonDelegatingQueryInterface(riid, ppv);
22. }

```

这个NonDelegatingQueryInterface()的写法确实够特别的，不过其作用还是一样的：根据不同的REFIID，获得不同的接口指针。在这里就不多说了。

再看一下Load()函数


```

1. // IFileSourceFilter
2. // 打开
3. STDMETHODIMP CLAVSplitter::Load(LPCOLESTR pszFileName, const AM_MEDIA_TYPE * pmt)
4. {
5.     CheckPointer(pszFileName, E_POINTER);
6.
7.     m_bPlaybackStarted = FALSE;
8.
9.     m_fileName = std::wstring(pszFileName);
10.
11.     HRESULT hr = S_OK;
12.     SAFE_DELETE(m_pDemuxer);
13.     LPWSTR extension = PathFindExtensionW(pszFileName);
14.
15.     DbgLog((LOG_TRACE, 10, L"::Load(): Opening file '%s' (extension: %s)", pszFileName, extension));
16.
17.     // BDMV uses the BD demuxer, everything else LAVF
18.     if (_wcsicmp(extension, L".bdmv") == 0 || _wcsicmp(extension, L".mpls") == 0) {
19.         m_pDemuxer = new CBDDemuxer(this, this);
20.     } else {
21.         m_pDemuxer = new CLAVFDemuxer(this, this);
22.     }
23.     //打开
24.     if(FAILED(hr = m_pDemuxer->Open(pszFileName))) {
25.         SAFE_DELETE(m_pDemuxer);
26.         return hr;
27.     }
28.     m_pDemuxer->AddRef();
29.
30.     return InitDemuxer();
31. }

```

在这里我们要注意CLAVSplitter的一个变量：m_pDemuxer。这是一个指向 CBaseDemuxer的指针。因此在这里CLAVSplitter实际上调用了 CBaseDemuxer中的方法。

从代码中的逻辑我们可以看出：

1.寻找文件后缀

2.当文件后缀是：".bdmv"或者".mpls"的时候，m_pDemuxer指向一个CBDDemuxer（我推测这代表目标文件是蓝光文件什么的），其他情况下m_pDemuxer指向一个CLAVFDemuxer。

3.然后m_pDemuxer会调用Open()方法。

4.最后会调用一个InitDemuxer()方法。

在这里我们应该看看m_pDemuxer->Open()这个方法里面有什么。我们先考虑m_pDemuxer指向CLAVFDemuxer的情况。

```

1. // Demuxer Functions
2. // 打开（就是一个封装）
3. STDMETHODIMP CLAVFDemuxer::Open(LPCOLESTR pszFileName)
4. {
5.     return OpenInputStream(NULL, pszFileName, NULL, TRUE);
6. }

```

发现是一层封装，于是果断决定层层深入。

[cpp]  

```
1. //实际的打开,使用FFMPEG
2. STDMETHODIMP CLAVFDemuxer::OpenInputStream(AVIOContext *byteContext, LPCOLESTR pszFileName, const char *format, BOOL bForce)
3. {
4.     CAutoLock lock(m_pLock);
5.     HRESULT hr = S_OK;
6.
7.     int ret; // return code from avformat functions
8.
9.     // Convert the filename from wchar to char for avformat
10.    char fileName[4100] = {0};
11.    if (pszFileName) {
12.        ret = WideCharToMultiByte(CP_UTF8, 0, pszFileName, -1, fileName, 4096, NULL, NULL);
13.    }
14.
15.    if (_strnicmp("mms:", fileName, 4) == 0) {
16.        memmove(fileName+1, fileName, strlen(fileName));
17.        memcpy(fileName, "mmsh", 4);
18.    }
19.
20.    AVIOInterruptCB cb = {avio_interrupt_cb, this};
21.
22.    trynoformat:
23.        // Create the avformat_context
24.        // FFMPEG中的函数
25.        m_avFormat = avformat_alloc_context();
26.        m_avFormat->pb = byteContext;
27.        m_avFormat->interrupt_callback = cb;
28.
29.        if (m_avFormat->pb)
30.            m_avFormat->flags |= AVFMT_FLAG_CUSTOM_IO;
31.
32.        LPWSTR extension = pszFileName ? PathFindExtensionW(pszFileName) : NULL;
33.
34.        AVInputFormat *inputFormat = NULL;
35.        //如果指定了格式
36.        if (format) {
37.            //查查有木有
38.            inputFormat = av_find_input_format(format);
39.        } else if (pszFileName) {
40.            LPWSTR extension = PathFindExtensionW(pszFileName);
41.            for (int i = 0; i < countof(wszImageExtensions); i++) {
42.                if (_wcsicmp(extension, wszImageExtensions[i]) == 0) {
43.                    if (byteContext) {
44.                        inputFormat = av_find_input_format("image2pipe");
45.                    } else {
46.                        inputFormat = av_find_input_format("image2");
47.                    }
48.                    break;
49.                }
50.            }
51.            for (int i = 0; i < countof(wszBlockedExtensions); i++) {
52.                if (_wcsicmp(extension, wszBlockedExtensions[i]) == 0) {
53.                    goto done;
54.                }
55.            }
56.        }
57.
58.        // Disable loading of external mkv segments, if required
59.        if (!m_pSettings->GetLoadMatroskaExternalSegments())
60.            m_avFormat->flags |= AVFMT_FLAG_NOEXTERNAL;
61.
62.        m_timeOpening = time(NULL);
63.        //实际的打开
64.        ret = avformat_open_input(&m_avFormat, fileName, inputFormat, NULL);
65.        //出错了
66.        if (ret < 0) {
67.            DbgLog((LOG_ERROR, 0, TEXT("::OpenInputStream(): avformat_open_input failed (%d)", ret)));
68.            if (format) {
69.                DbgLog((LOG_ERROR, 0, TEXT(" -> trying again without specific format")));
70.                format = NULL;
71.                //实际的关闭
72.                avformat_close_input(&m_avFormat);
73.                goto trynoformat;
74.            }
75.            goto done;
76.        }
77.        DbgLog((LOG_TRACE, 10, TEXT("::OpenInputStream(): avformat_open_input opened file of type '%S' (took %I64d seconds)", m_avFormat->iFormat->name, time(NULL) - m_timeOpening));
78.        m_timeOpening = 0;
79.        //初始化AVFormat
80.        CHECK_HR(hr = InitAVFormat(pszFileName, bForce));
81.
82.        return S_OK;
83.    done:
84.        CleanupAVFormat();
85.        return E_FAIL;
86.    }
```

看到这个函数，立马感受到了一种“拨云见日”的感觉。看到了很多FFmpeg的API函数。最重要的依据当属avformat_open_input()了，通过这个函数，打开了实际的文件。如果出现错误，则调用avformat_close_input()进行清理。

最后，还调用了InitAVFormat()函数：

```
[cpp]
1. //初始化AVFormat
2. STDMETHODIMP CLAVDemuxer::InitAVFormat(LPCOLESTR pszFileName, BOOL bForce)
3. {
4.     HRESULT hr = S_OK;
5.     const char *format = NULL;
6.     //获取InputFormat信息 (, 短名称, 长名称)
7.     lavf_get_ifformat_infos(m_avFormat->ifformat, &format, NULL);
8.     if (!bForce && (!format || !m_pSettings->IsFormatEnabled(format))) {
9.         DbgLog((LOG_TRACE, 20, L"::InitAVFormat() - format of type '%S' disabled, failing", format ? format : m_avFormat->ifformat->name)
10.    );
11.     return E_FAIL;
12.    }
13.
14.    m_pszInputFormat = format ? format : m_avFormat->ifformat->name;
15.
16.    m_bVC1SeenTimestamp = FALSE;
17.
18.    LPWSTR extension = pszFileName ? PathFindExtensionW(pszFileName) : NULL;
19.
20.    m_bMatroska = (_strnicmp(m_pszInputFormat, "matroska", 8) == 0);
21.    m_bOgg = (_strnicmp(m_pszInputFormat, "ogg", 3) == 0);
22.    m_bAVI = (_strnicmp(m_pszInputFormat, "avi", 3) == 0);
23.    m_bMPEGTS = (_strnicmp(m_pszInputFormat, "mpegts", 6) == 0);
24.    m_bMPEGPS = (_stricmp(m_pszInputFormat, "mpeg") == 0);
25.    m_bRM = (_stricmp(m_pszInputFormat, "rm") == 0);
26.    m_bPMP = (_stricmp(m_pszInputFormat, "pmp") == 0);
27.    m_bMP4 = (_stricmp(m_pszInputFormat, "mp4") == 0);
28.
29.    m_bTSDiscont = m_avFormat->ifformat->flags & AVFMT_TS_DISCONT;
30.
31.    WCHAR szProt[24] = L"file";
32.    if (pszFileName) {
33.        DWORD dwNumChars = 24;
34.        hr = UrlGetPart(pszFileName, szProt, &dwNumChars, URL_PART_SCHEME, 0);
35.        if (SUCCEEDED(hr) && dwNumChars && (_wcsicmp(szProt, L"file") != 0)) {
36.            m_avFormat->flags |= AVFMT_FLAG_NETWORK;
37.            DbgLog((LOG_TRACE, 10, TEXT("::InitAVFormat(): detected network protocol: %s"), szProt));
38.        }
39.    }
40.
41.    // TODO: make both durations below configurable
42.    // decrease analyze duration for network streams
43.    if (m_avFormat->flags & AVFMT_FLAG_NETWORK || (m_avFormat->flags & AVFMT_FLAG_CUSTOM_IO && !m_avFormat->pb->seekable)) {
44.        // require at least 0.2 seconds
45.        m_avFormat->max_analyze_duration = max(m_pSettings->GetNetworkStreamAnalysisDuration() * 1000, 200000);
46.    } else {
47.        // And increase it for mpeg-ts/ps files
48.        if (m_bMPEGTS || m_bMPEGPS)
49.            m_avFormat->max_analyze_duration = 10000000;
50.    }
51.
52.    av_opt_set_int(m_avFormat, "correct_ts_overflow", !m_pBluRay, 0);
53.
54.    if (m_bMatroska)
55.        m_avFormat->flags |= AVFMT_FLAG_KEEP_SIDE_DATA;
56.
57.    m_timeOpening = time(NULL);
58.    //获取媒体流信息
59.    int ret = avformat_find_stream_info(m_avFormat, NULL);
60.    if (ret < 0) {
61.        DbgLog((LOG_ERROR, 0, TEXT("::InitAVFormat(): av_find_stream_info failed (%d)", ret));
62.        goto done;
63.    }
64.    DbgLog((LOG_TRACE, 10, TEXT("::InitAVFormat(): avformat_find_stream_info finished, took %I64d seconds"), time(NULL) - m_timeOpening));
65.
66.    m_timeOpening = 0;
67.
68.    // Check if this is a m2ts in a BD structure, and if it is, read some extra stream properties out of the CLPI files
69.    if (m_pBluRay) {
70.        m_pBluRay->ProcessClipLanguages();
71.    } else if (pszFileName && m_bMPEGTS) {
72.        CheckBDM2TSCPLI(pszFileName);
73.    }
74.
75.    SAFE_CO_FREE(m_stOrigParser);
76.    m_stOrigParser = (enum AVStreamParseType *)CoTaskMemAlloc(m_avFormat->n_buffers * sizeof(enum AVStreamParseType));
77.    if (!m_stOrigParser)
78.        return E_OUTOFMEMORY;
79.
80.    for(unsigned int idx = 0; idx < m_avFormat->n_buffers; ++idx) {
81.        AVStream *st = m_avFormat->streams[idx];
82.
83.        // Disable full stream parsing for these formats
84.        if (st->need_parsing == AVSTREAM_PARSE_FULL) {
```

```

83.         if (st->codec->codec_id == AV_CODEC_ID_DVB_SUBTITLE) {
84.             st->need_parsing = AVSTREAM_PARSE_NONE;
85.         }
86.     }
87.
88.     if (m_b0gg && st->codec->codec_id == AV_CODEC_ID_H264) {
89.         st->need_parsing = AVSTREAM_PARSE_FULL;
90.     }
91.
92.     // Create the parsers with the appropriate flags
93.     init_parser(m_avFormat, st);
94.     UpdateParserFlags(st);
95.
96. #ifdef DEBUG
97.     AVProgram *streamProg = av_find_program_from_stream(m_avFormat, NULL, idx);
98.     DbgLog((LOG_TRACE, 30, L"Stream %d (pid %d) - program: %d, codec: %S; parsing: %S;", idx, st->id, streamProg ? streamProg->pmt_p
99. id : -1, avcodec_get_name(st->codec->codec_id), lavf_get_parsing_string(st->need_parsing)));
100. #endif
101.     m_stOrigParser[idx] = st->need_parsing;
102.
103.     if ((st->codec->codec_id == AV_CODEC_ID_DTS && st->codec->codec_tag == 0xA2)
104.         || (st->codec->codec_id == AV_CODEC_ID_EAC3 && st->codec->codec_tag == 0xA1))
105.         st->disposition |= LAVF_DISPOSITION_SECONDARY_AUDIO;
106.
107.     UpdateSubStreams();
108.
109.     if (st->codec->codec_type == AVMEDIA_TYPE_ATTACHMENT && (st->codec->codec_id == AV_CODEC_ID_TTF || st->codec->codec_id == AV_COD
110. EC_ID_OTF)) {
111.         if (!m_pFontInstaller) {
112.             m_pFontInstaller = new CFontInstaller();
113.         }
114.         m_pFontInstaller->InstallFont(st->codec->extradata, st->codec->extradata_size);
115.     }
116.
117.     CHECK_HR(hr = CreateStreams());
118.
119.     return S_OK;
120. done:
121.     //关闭输入
122.     CleanupAVFormat();
123.     return E_FAIL;
124. }

```

该函数通过avformat_find_stream_info()等获取到流信息，这里就不多说了。

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/12711723>

文章标签： [LAVFilter](#) [LAVSplitter](#) [源代码](#) [分析](#)

个人分类： [LAV Filter](#)

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