原 XBMC源代码分析 7：视频播放器（dvdplayer）-输入流（以libRTMP为例）

2014年01月10日 01:05:43 阅读数：8166

前文分析了XBMC的基本结构：

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

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

XBMC源代码简析 5：视频播放器（dvdplayer）-解复用器（以ffmpeg为例）

XBMC源代码分析 6：视频播放器（dvdplayer）-文件头（以ffmpeg为例）

本文我们分析XBMC中视频播放器（dvdplayer）中的输入流部分。由于输入流种类很多，因此以RTMP输入流为例进行分析。

XBMC中输入流部分文件目录结构如下图所示。

从目录中文件的名称我们可以看出，XBMC支持多种输入方式：File，HTSP，HTTP，RTMP等等。在这里我们看看RTMP部分的源代码。对应DVDInputStreamRTMP.h和DVDInputStreamRTMP.cpp

先来看看DVDInputStreamRTMP.h

```cpp
/*
 * 雷霄骅
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 *
 */
//如果有libRTMP
#ifdef HAS_LIBRTMP

#include "DVDInputStream.h"
#include "DllLibRTMP.h"
//支持RTMP输入流的类，继承CDVDInputStream
class CDVDInputStreamRTMP
  : public CDVDInputStream
  , public CDVDInputStream::ISeekTime
{
public:
  CDVDInputStreamRTMP();
  virtual ~CDVDInputStreamRTMP();
  virtual bool    Open(const char* strFile, const std::string &content);//打开
  virtual void    Close();//关闭
  virtual int     Read(uint8_t* buf, int buf_size);//读取
  virtual int64_t Seek(int64_t offset, int whence);//跳转到
  bool            SeekTime(int iTimeInMsec);
  virtual bool Pause(double dTime);//暂停
  virtual bool    IsEOF();
  virtual int64_t GetLength();

  CCriticalSection m_RTMPSection;

protected:
  bool       m_eof;
  bool       m_bPaused;
  char*      m_sStreamPlaying;
  std::vector<CStdString> m_optionvalues;

  RTMP       *m_rtmp;
  DllLibRTMP m_libRTMP;
};

#endif
```

该类中包含了Open()，Close()，Read()，Seek()，Pause() 这类的方法。实现了对RTMP协议的各种操作。这些方法都是CDVDInputStreamRTMP父类CDVDInputStream中的方法。可以看一下CDVDInputStream的定义，就知道了。

```cpp
//输入流类
class CDVDInputStream
```

```
{
public:
  class IChannel
  {
    public:
    virtual ~IChannel() {};
    virtual bool NextChannel(bool preview = false) = 0;
    virtual bool PrevChannel(bool preview = false) = 0;
    virtual bool SelectChannelByNumber(unsigned int channel) = 0;
    virtual bool SelectChannel(const PVR::CPVRChannel &channel) { return false; };
    virtual bool GetSelectedChannel(PVR::CPVRChannelPtr&) { return false; };
    virtual bool UpdateItem(CFileItem& item) = 0;
    virtual bool CanRecord() = 0;
    virtual bool IsRecording() = 0;
    virtual bool Record(bool bOnOff) = 0;
    virtual bool CanPause() = 0;
    virtual bool CanSeek() = 0;
  };

  class IDisplayTime
  {
    public:
    virtual ~IDisplayTime() {};
    virtual int GetTotalTime() = 0;
    virtual int GetTime() = 0;
  };

  class ISeekTime
  {
    public:
    virtual ~ISeekTime() {};
    virtual bool SeekTime(int ms) = 0;
  };

  class IChapter
  {
    public:
    virtual ~IChapter() {};
    virtual int  GetChapter() = 0;
    virtual int  GetChapterCount() = 0;
    virtual void GetChapterName(std::string& name) = 0;
    virtual bool SeekChapter(int ch) = 0;
  };

  class IMenus
  {
    public:
    virtual ~IMenus() {};
    virtual void ActivateButton() = 0;
    virtual void SelectButton(int iButton) = 0;
    virtual int  GetCurrentButton() = 0;
    virtual int  GetTotalButtons() = 0;
    virtual void OnUp() = 0;
    virtual void OnDown() = 0;
    virtual void OnLeft() = 0;
    virtual void OnRight() = 0;
    virtual void OnMenu() = 0;
    virtual void OnBack() = 0;
    virtual void OnNext() = 0;
    virtual void OnPrevious() = 0;
    virtual bool OnMouseMove(const CPoint &point) = 0;
    virtual bool OnMouseClick(const CPoint &point) = 0;
    virtual bool IsInMenu() = 0;
    virtual void SkipStill() = 0;
    virtual double GetTimeStampCorrection() = 0;
    virtual bool GetState(std::string &xmlstate) = 0;
    virtual bool SetState(const std::string &xmlstate) = 0;

  };

  class ISeekable
  {
    public:
    virtual ~ISeekable() {};
    virtual bool CanSeek()  = 0;
    virtual bool CanPause() = 0;
  };

  enum ENextStream
  {
    NEXTSTREAM_NONE,
    NEXTSTREAM_OPEN,
    NEXTSTREAM_RETRY,
  };

  CDVDInputStream(DVDStreamType m_streamType);
  virtual ~CDVDInputStream();
  virtual bool Open(const char* strFileName, const std::string& content);//打开
  virtual void Close() = 0;//关闭
  virtual int Read(uint8_t* buf, int buf_size) = 0;//读取
  virtual int64_t Seek(int64_t offset, int whence) = 0;//跳转
```

```cpp
  virtual bool Pause(double dTime) = 0;//暂停
  virtual int64_t GetLength() = 0;
  virtual std::string& GetContent() { return m_content; };
  virtual std::string& GetFileName() { return m_strFileName; }
  virtual CURL &GetURL() { return m_url; }
  virtual ENextStream NextStream() { return NEXTSTREAM_NONE; }
  virtual void Abort() {}
  virtual int GetBlockSize() { return 0; }
  virtual void ResetScanTimeout(unsigned int iTimeoutMs) { }

  /*! \brief Indicate expected read rate in bytes per second.
   *  This could be used to throttle caching rate. Should
   *  be seen as only a hint
   */
  virtual void SetReadRate(unsigned rate) {}

  /*! \brief Get the cache status
   \return true when cache status was succesfully obtained
   */
  virtual bool GetCacheStatus(XFILE::SCacheStatus *status) { return false; }

  bool IsStreamType(DVDStreamType type) const { return m_streamType == type; }
  virtual bool IsEOF() = 0;
  virtual BitstreamStats GetBitstreamStats() const { return m_stats; }

  void SetFileItem(const CFileItem& item);

protected:
  DVDStreamType m_streamType;
  std::string m_strFileName;
  CURL m_url;
  BitstreamStats m_stats;
  std::string m_content;
  CFileItem m_item;
};
```

回到CDVDInputStreamRTMP类本身。可以看一下Open()，Close()，Read()，Seek()，Pause()这些方法的函数体。这些方方通过调用libRTMP中相应的方法，完成了对RTMP流媒体的各种操作。

```cpp
/*
 * 雷霄骅
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 *
 */
//打开
bool CDVDInputStreamRTMP::Open(const char* strFile, const std::string& content)
{
  if (m_sStreamPlaying)
  {
    free(m_sStreamPlaying);
    m_sStreamPlaying = NULL;
  }

  if (!CDVDInputStream::Open(strFile, "video/x-flv"))
    return false;

  CSingleLock lock(m_RTMPSection);

  // libRTMP can and will alter strFile, so take a copy of it
  m_sStreamPlaying = (char*)calloc(strlen(strFile)+1,sizeof(char));
  strcpy(m_sStreamPlaying,strFile);
  //libRTMP中的设置URL
  if (!m_libRTMP.SetupURL(m_rtmp, m_sStreamPlaying))
    return false;

  // SetOpt and SetAVal copy pointers to the value. librtmp doesn't use the values until the Connect() call,
  // so value objects must stay allocated until then. To be extra safe, keep the values around until Close(),
  // in case librtmp needs them again.
  m_optionvalues.clear();
  for (int i=0; options[i].name; i++)
  {
    CStdString tmp = m_item.GetProperty(options[i].name).asString();
    if (!tmp.empty())
    {
      m_optionvalues.push_back(tmp);
      AVal av_tmp;
      SetAVal(av_tmp, m_optionvalues.back());
      m_libRTMP.SetOpt(m_rtmp, &options[i].key, &av_tmp);
    }
  }
  //建立RTMP链接中的NetConnection和NetStream
  if (!m_libRTMP.Connect(m_rtmp, NULL) || !m_libRTMP.ConnectStream(m_rtmp, 0))
    return false;

```

```
  m_eof = false;

  return true;
}
//关闭
// close file and reset everything
void CDVDInputStreamRTMP::Close()
{
  CSingleLock lock(m_RTMPSection);
  CDVDInputStream::Close();
  //关闭连接
  m_libRTMP.Close(m_rtmp);

  m_optionvalues.clear();
  m_eof = true;
  m_bPaused = false;
}
//读取
int CDVDInputStreamRTMP::Read(uint8_t* buf, int buf_size)
{//读取
  int i = m_libRTMP.Read(m_rtmp, (char *)buf, buf_size);
  if (i < 0)
    m_eof = true;

  return i;
}
//跳转到
int64_t CDVDInputStreamRTMP::Seek(int64_t offset, int whence)
{
  if (whence == SEEK_POSSIBLE)
    return 0;
  else
    return -1;
}
//暂停
bool CDVDInputStreamRTMP::Pause(double dTime)
{
  CSingleLock lock(m_RTMPSection);

  m_bPaused = !m_bPaused;

  CLog::Log(LOGNOTICE, "RTMP Pause %s requested", m_bPaused ? "TRUE" : "FALSE");

  m_libRTMP.Pause(m_rtmp, m_bPaused);

  return true;
}
```


文章标签： xbmc librtmp 源代码 播放器 输入

个人分类： XBMC libRTMP

所属专栏： 开源多媒体项目源代码分析