原 MediaInfo源代码分析 2：API函数

2013年10月08日 19:58:33　阅读数：6279

==========

MediaInfo源代码分析系列文章列表：

==========

本文主要分析MediaInfo的API函数。它的API函数位于MediaInfo.h文件中的一个叫做MediaInfo的类中。

该类如下所示，部分重要的方法已经加上了注释：

```cpp
//MediaInfo类
class MEDIAINFO_EXP MediaInfo
{
public :
    //Constructor/Destructor
    MediaInfo ();
    ~MediaInfo ();
    //File
        /// Open a file and collect information about it (technical information and tags)
        /// @brief Open a file
        /// @param File_Name Full name of file to open
        /// @retval 0 File not opened
        /// @retval 1 File opened
    //打开文件
    size_t Open (const String &File_Name);
        /// Open a Buffer (Begin and end of the stream) and collect information about it (technical information and tags)
        /// @brief Open a buffer
        /// @param Begin First bytes of the buffer
        /// @param Begin_Size Size of Begin
        /// @param End Last bytes of the buffer
        /// @param End_Size Size of End
        /// @param File_Size Total size of the file
        /// @retval 0 File not opened
        /// @retval 1 File opened
    //打开一段内存！
    size_t Open (const ZenLib::int8u* Begin, size_t Begin_Size, const ZenLib::int8u* End=NULL, size_t End_Size=0, ZenLib::int64u File_Size=0);
        /// Open a stream and collect information about it (technical information and tags)
        /// @brief Open a stream (Init)
        /// @param File_Size Estimated file size
        /// @param File_Offset Offset of the file (if we don't have the beginning of the file)
        /// @retval 0 File not opened
        /// @retval 1 File opened
    //打开一个流和收集的关于它的信息
    size_t Open_Buffer_Init (ZenLib::int64u File_Size=(ZenLib::int64u)-1, ZenLib::int64u File_Offset=0);
        /// Open a stream and collect information about it (technical information and tags)
        /// @brief Open a stream (Continue)
        /// @param Buffer pointer to the stream
        /// @param Buffer_Size Count of bytes to read
        /// @return a bitfield \n
        ///         bit 0: Is Accepted  (format is known)
        ///         bit 1: Is Filled    (main data is collected)
        ///         bit 2: Is Updated   (some data have beed updated, example: duration for a real time MPEG-TS stream)
        ///         bit 3: Is Finalized (No more data is needed, will not use further data)
        ///         bit 4-15: Reserved
        ///         bit 16-31: User defined
    //打开一个流和收集的关于它的信息
    size_t Open_Buffer_Continue (const ZenLib::int8u* Buffer, size_t Buffer_Size);
        /// Open a stream and collect information about it (technical information and tags)
        /// @brief Open a stream (Get the needed file Offset)
        /// @return the needed offset of the file \n
        ///         File size if no more bytes are needed
    ZenLib::int64u Open_Buffer_Continue_GoTo_Get ();
        /// Open a stream and collect information about it (technical information and tags)
        /// @brief Open a stream (Finalize)
        /// @retval 0 failed
        /// @retval 1 succeed
    //打开一个流和收集的关于它的信息
    size_t Open_Buffer_Finalize ();
        /// If Open() is used in "PerPacket" mode, parse only one packet and return
```

```
        /// @brief Read one packet (if "PerPacket" mode is set)
        /// @return a bitfield \n
        ///         bit 0: A packet was read
    size_t Open_NextPacket ();
        /// (NOT IMPLEMENTED YET) Save the file opened before with Open() (modifications of tags)
        /// @brief (NOT IMPLEMENTED YET) Save the file
        /// @retval 0 failed
        /// @retval 1 suceed
    size_t Save ();
        /// Close a file opened before with Open() (without saving)
        /// @brief Close a file
        /// @warning without have saved before, modifications are lost
    void Close ();

    //General information
        /// Get all details about a file in one string
        /// @brief Get all details about a file
        /// @param Reserved Reserved, do not use
        /// @pre You can change default presentation with Inform_Set()
        /// @return Text with information about the file
    //返回文件信息
    String Inform (size_t Reserved=0);

    //Get
        /// Get a piece of information about a file (parameter is an integer)
        /// @brief Get a piece of information about a file (parameter is an integer)
        /// @param StreamKind Kind of stream (general, video, audio...)
        /// @param StreamNumber Stream number in Kind of stream (first, second...)
        /// @param Parameter Parameter you are looking for in the stream (Codec, width, bitrate...), in integer format (first paramet
 second parameter...)
        /// @param InfoKind Kind of information you want about the parameter (the text, the measure, the help...)
        /// @return a string about information you search \n
        ///         an empty string if there is a problem
    //获取一部分文件的信息（参数是一个整数）
    String Get (stream_t StreamKind, size_t StreamNumber, size_t Parameter, info_t InfoKind=Info_Text);
        /// Get a piece of information about a file (parameter is a string)
        /// @brief Get a piece of information about a file (parameter is a string)
        /// @param StreamKind Kind of stream (general, video, audio...)
        /// @param StreamNumber Stream number in Kind of stream (first, second...)
        /// @param Parameter Parameter you are looking for in the stream (Codec, width, bitrate...), in string format ("Codec", "Widt
..) \n
        ///        See MediaInfo::Option("Info_Parameters") to have the full list
        /// @param InfoKind Kind of information you want about the parameter (the text, the measure, the help...)
        /// @param SearchKind Where to look for the parameter
        /// @return a string about information you search \n
        ///         an empty string if there is a problem
    //获取一部分文件的信息（参数是一个字符串）
    String Get (stream_t StreamKind, size_t StreamNumber, const String &Parameter, info_t InfoKind=Info_Text, info_t SearchKind=Info_
ame);

    //Set
        /// (NOT IMPLEMENTED YET) Set a piece of information about a file (parameter is an integer)
        /// @brief (NOT IMPLEMENTED YET) Set a piece of information about a file (parameter is an int)
        /// @warning Not yet implemented, do not use it
        /// @param ToSet Piece of information
        /// @param StreamKind Kind of stream (general, video, audio...)
        /// @param StreamNumber Stream number in Kind of stream (first, second...)
        /// @param Parameter Parameter you are looking for in the stream (Codec, width, bitrate...), in integer format (first paramet
 second parameter...)
        /// @param OldValue The old value of the parameter \n if OldValue is empty and ToSet is filled: tag is added \n if OldValue i
illed and ToSet is filled: tag is replaced \n if OldValue is filled and ToSet is empty: tag is deleted
        /// @retval >0 succeed
        /// @retval 0 failed
    size_t Set (const String &ToSet, stream_t StreamKind, size_t StreamNumber, size_t Parameter, const String &OldValue=String());
        /// (NOT IMPLEMENTED YET) Set a piece of information about a file (parameter is a string)
        /// @warning Not yet implemented, do not use it
        /// @brief (NOT IMPLEMENTED YET) Set information about a file (parameter is a string)
        /// @param ToSet Piece of information
        /// @param StreamKind Kind of stream (general, video, audio...)
        /// @param StreamNumber Stream number in Kind of stream (first, second...)
        /// @param Parameter Parameter you are looking for in the stream (Codec, width, bitrate...), in string format
        /// @param OldValue The old value of the parameter \n if OldValue is empty and ToSet is filled: tag is added \n if OldValue i
illed and ToSet is filled: tag is replaced \n if OldValue is filled and ToSet is empty: tag is deleted
        /// @retval >0 succeed
        /// @retval 0 failed
    size_t Set (const String &ToSet, stream_t StreamKind, size_t StreamNumber, const String &Parameter, const String &OldValue=String
());

    //Output_Buffered
        /// Output the written size when "File_Duplicate" option is used.
        /// @brief Output the written size when "File_Duplicate" option is used.
        /// @param Value The unique name of the duplicated stream (begin with "memory://")
        /// @return The size of the used buffer
    size_t Output_Buffer_Get (const String &Value);
        /// Output the written size when "File_Duplicate" option is used.
        /// @brief Output the written size when "File_Duplicate" option is used.
        /// @param Pos The order of calling
        /// @return The size of the used buffer
    size_t Output_Buffer_Get (size_t Pos);

    //Info
```

```
        /// Configure or get information about MediaInfoLib
        /// @param Option The name of option
        /// @param Value The value of option
        /// @return Depend of the option: by default "" (nothing) means No, other means Yes
        /// @post Known options are: \n
        ///       * (NOT IMPLEMENTED YET) "BlockMethod": Configure when Open Method must return (default or not command not understoo
"1") \n
        ///                 "0": Immediatly \n
        ///                 "1": After geting local information \n
        ///                 "2": When user interaction is needed, or whan Internet information is get
        ///       * "Complete": For debug, configure if MediaInfoLib::Inform() show all information (doesn't care of InfoOption_NoSho
ag): shows all information if true, shows only useful for user information if false (No by default)\n
        ///       * "Complete_Get": return the state of "Complete" \n
        ///       * "Language": Configure language (default language, and this object); Value is Description of language (format: "Co
n1;Colum2\n...) \n
        ///                 Column 1: Unique name ("Bytes", "Title") \n
        ///                 Column 2: translation ("Octets", "Titre") \n
        ///       * "Language_Get": Get the language file in memory
        ///       * "Language_Update": Configure language of this object only (for optimisation); Value is Description of language (f
at: "Column1;Colum2\n...) \n
        ///                 Column 1: Unique name ("Bytes", "Title") \n
        ///                 Column 2: translation ("Octets", "Titre") \n
        ///       * "Inform": Configure custom text, See MediaInfoLib::Inform() function; Description of views (format: "Column1;Colu
..) \n
        ///                 Column 1: code (11 lines: "General", "Video", "Audio", "Text", "Other", "Begin", "End", "Page_Begin", "Pa
Middle", "Page_End") \n
        ///                 Column 2: The text to show (exemple: "Audio: %FileName% is at %BitRate/String%") \n
        ///       * "ParseUnknownExtensions": Configure if MediaInfo parse files with unknown extension\n
        ///       * "ParseUnknownExtensions_Get": Get if MediaInfo parse files with unknown extension\n
        ///       * "ShowFiles": Configure if MediaInfo keep in memory files with specific kind of streams (or no streams); Value is
cription of components (format: "Column1;Colum2\n...) \n\n
        ///                 Column 1: code (available: "Nothing" for unknown format, "VideoAudio" for at least 1 video and 1 audio, "
eoOnly" for video streams only, "AudioOnly", "TextOnly") \n
        ///                 Column 2: "" (nothing) not keeping, other for keeping
        ///       * (NOT IMPLEMENTED YET) "TagSeparator": Configure the separator if there are multiple same tags (" | " by default)\

        ///       * (NOT IMPLEMENTED YET) "TagSeparator_Get": return the state of "TagSeparator" \n
        ///       * (NOT IMPLEMENTED YET) "Internet": Authorize Internet connection (Yes by default)
        ///       * (NOT IMPLEMENTED YET) "Internet_Title_Get": When State=5000, give all possible titles for this file (one per line
n
        ///                 Form: Author TagSeparator Title TagSeparator Year\n...
        ///       * (NOT IMPLEMENTED YET) "Internet_Title_Set": Set the Good title (same as given by Internet_Title_Get) \n
        ///                 Form: Author TagSeparator Title TagSeparator Year
        ///       * "Info_Parameters": Information about what are known unique names for parameters \n
        ///       * "Info_Parameters_CSV": Information about what are known unique names for parameters, in CSV format \n
        ///       * "Info_Codecs": Information about which codec is known \n
        ///       * "Info_Version": Information about the version of MediaInfoLib
        ///       * "Info_Url": Information about where to find the last version
    //配置
    String        Option (const String &Option, const String &Value=String());
        /// Configure or get information about MediaInfoLib
        /// @param Option The name of option
        /// @param Value The value of option
        /// @return Depend of the option: by default "" (nothing) means No, other means Yes
        /// @post Known options are: See MediaInfo::Option()
    static String Option_Static (const String &Option, const String &Value=String());
        /// @brief (NOT IMPLEMENTED YET) Get the state of the library
        /// @retval <1000 No information is available for the file yet
        /// @retval >=1000_<5000 Only local (into the file) information is available, getting Internet information (titles only) is n
inished yet
        /// @retval 5000 (only if Internet connection is accepted) User interaction is needed (use Option() with "Internet_Title_Get"
n
        ///             Warning: even there is only one possible, user interaction (or the software) is needed
        /// @retval >5000<=10000 Only local (into the file) information is available, getting Internet information (all) is no finish
yet
        /// @retval <10000 Done
    size_t                State_Get ();
        /// @brief Count of streams of a stream kind (StreamNumber not filled), or count of piece of information in this stream
        /// @param StreamKind Kind of stream (general, video, audio...)
        /// @param StreamNumber Stream number in this kind of stream (first, second...)
        /// @return The count of fields for this stream kind / stream number if stream number is provided, else the count of streams
 this stream kind
    size_t                Count_Get (stream_t StreamKind, size_t StreamNumber=(size_t)-1);

private :
    MediaInfo_Internal* Internal;

    //Constructor
    MediaInfo (const MediaInfo&);                           // Prevent copy-construction
    MediaInfo& operator=(const MediaInfo&);                 // Prevent assignment
};
```

由代码可见，MediaInfo实际上不仅仅可以读取一个特定路径的文件【Open (const String &File_Name);】

，而且可以读取一块内存中的数据【Open (const ZenLib::int8u* Begin, size_t Begin_Size, const ZenLib::int8u* End=NULL, size_t End_Size=0, Z enLib::int64u File_Size=0)】（这个函数目前还没有使用过）。

如果想要获取完整的信息，可以使用【Inform (size_t Reserved=0)】，并且使用【Option (const String &Option, const String &Value=String())】进行配置。

如果只想获得特定的信息，可以使用【Get (stream_t StreamKind, size_t StreamNumber, size_t Parameter, info_t InfoKind=Info_Text)】。

具体使用方法参见 http://blog.csdn.net/leixiaohua1020/article/details/11902195 。

文章标签： mediainfo api 源代码

个人分类： MediaInfo

所属专栏： 开源多媒体项目源代码分析