

## 原 RTMPDump (libRTMP) 源代码分析 2：解析RTMP地址——RTMP\_ParseURL()

2013年10月22日 21:04:57 阅读数：16626

=====

RTMPdump(libRTMP) 源代码分析系列文章：

[RTMPdump 源代码分析 1：main\(\)函数](#)

[RTMPDump \(libRTMP\) 源代码分析2：解析RTMP地址——RTMP\\_ParseURL\(\)](#)

[RTMPdump \(libRTMP\) 源代码分析3：AMF编码](#)

[RTMPdump \(libRTMP\) 源代码分析4：连接第一步——握手 \(HandShake\)](#)

[RTMPdump \(libRTMP\) 源代码分析5：建立一个流媒体连接 \(NetConnection部分\)](#)

[RTMPdump \(libRTMP\) 源代码分析6：建立一个流媒体连接 \(NetStream部分 1\)](#)

[RTMPdump \(libRTMP\) 源代码分析7：建立一个流媒体连接 \(NetStream部分 2\)](#)

[RTMPdump \(libRTMP\) 源代码分析8：发送消息 \(Message\)](#)

[RTMPdump \(libRTMP\) 源代码分析9：接收消息 \(Message\) \(接收视音频数据\)](#)

[RTMPdump \(libRTMP\) 源代码分析10：处理各种消息 \(Message\)](#)

=====

### 函数调用结构图

RTMPDump (libRTMP)的整体的函数调用结构图如下图所示。



[单击查看大图](#)

### 详细分析

之前分析了一下RTMPDump的Main()函数，其中获取RTMP流媒体数据很重要的前提是RTMP的URL的解析。如果没有这一步，那程序在强大也是白搭。现在来解析一下这个函数吧：RTMP\_ParseURL()。

下面首先回顾一下RTMP的URL的格式：

rtmp://localhost/vod/mp4:sample1\_1500kbps.f4v

“://”之前的是使用的协议类型，可以是rtmp, rtmpt, rtmps等



之后是服务器地址

再之后是端口号（可以没有，默认1935）

在之后是application的名字，在这里是“vod”

最后是流媒体文件路径。

关于URL就不多说了，可以查看相关文档，下面贴上注释后的代码（整个parseurl.c）：

```
[cpp]  
1.  /*
2.  * 本文件主要包含了对输入URL的解析
3.  */
4.  #include "stdafx.h"
5.  #include <stdlib.h>
6.  #include <string.h>
7.
8.  #include <assert.h>
9.  #include <ctype.h>
10.
11. #include "rtmp_sys.h"
12. #include "log.h"
13.
14. /*解析URL，得到协议名称(protocol)，主机名称(host)，应用程序名称(app)
15.  *
```

```

16.  */
17. int RTMP_ParseURL(const char *url, int *protocol, AVal *host, unsigned int *port,
18. AVal *playpath, AVal *app)
19. {
20.     char *p, *end, *col, *ques, *slash;
21.
22.     RTMP_Log(RTMP_LOGDEBUG, "Parsing...");
23.
24.     *protocol = RTMP_PROTOCOL_RTMP;
25.     *port = 0;
26.     playpath->av_len = 0;
27.     playpath->av_val = NULL;
28.     app->av_len = 0;
29.     app->av_val = NULL;
30.
31.     /* 字符串解析 */
32.     /* 查找“://” */
33.     //函数原型:char *strstr(char *str1, char *str2);
34.     //功能:找出str2字符串在str1字符串中第一次出现的位置(不包括str2的串结束符)。
35.     //返回值:返回该位置的指针,如找不到,返回空指针。
36.     p = strstr((char *)url, "://");
37.     if(!p) {
38.         RTMP_Log(RTMP_LOGERROR, "RTMP URL: No :// in url!");
39.         return FALSE;
40.     }
41.     {
42.         //指针相减,返回“://”之前字符串长度len
43.         int len = (int)(p-url);
44.         //获取使用的协议
45.         //通过比较字符串的方法
46.         if(len == 4 && strncasecmp(url, "rtmp", 4)==0)
47.             *protocol = RTMP_PROTOCOL_RTMP;
48.         else if(len == 5 && strncasecmp(url, "rtmpt", 5)==0)
49.             *protocol = RTMP_PROTOCOL_RTMPT;
50.         else if(len == 5 && strncasecmp(url, "rtmps", 5)==0)
51.             *protocol = RTMP_PROTOCOL_RTMP;
52.         else if(len == 5 && strncasecmp(url, "rtmpe", 5)==0)
53.             *protocol = RTMP_PROTOCOL_RTMPE;
54.         else if(len == 5 && strncasecmp(url, "rtmf", 5)==0)
55.             *protocol = RTMP_PROTOCOL_RTMFP;
56.         else if(len == 6 && strncasecmp(url, "rtmpte", 6)==0)
57.             *protocol = RTMP_PROTOCOL_RTMPT;
58.         else if(len == 6 && strncasecmp(url, "rtmpts", 6)==0)
59.             *protocol = RTMP_PROTOCOL_RTMPTS;
60.         else {
61.             RTMP_Log(RTMP_LOGWARNING, "Unknown protocol!\n");
62.             goto parsehost;
63.         }
64.     }
65.
66.     RTMP_Log(RTMP_LOGDEBUG, "Parsed protocol: %d", *protocol);
67.
68. parsehost:
69.     //获取主机名称
70.     //跳过“://”
71.     p+=3;
72.
73.     /* 检查一下主机名 */
74.     if(*p==0) {
75.         RTMP_Log(RTMP_LOGWARNING, "No hostname in URL!");
76.         return FALSE;
77.     }
78.     //原型:char *strchr(const char *s,char c);
79.     //功能:查找字符串s中首次出现字符c的位置
80.     //说明:返回首次出现c的位置的指针,如果s中不存在c则返回NULL。
81.     end = p + strlen(p); //指向结尾的指针
82.     col = strchr(p, ':'); //指向冒号(第一个)的指针
83.     ques = strchr(p, '?'); //指向问号(第一个)的指针
84.     slash = strchr(p, '/'); //指向斜杠(第一个)的指针
85.
86.     {
87.         int hostlen;
88.         if(slash)
89.             hostlen = slash - p;
90.         else
91.             hostlen = end - p;
92.         if(col && col - p < hostlen)
93.             hostlen = col - p;
94.
95.         if(hostlen < 256) {
96.             host->av_val = p;
97.             host->av_len = hostlen;
98.             RTMP_Log(RTMP_LOGDEBUG, "Parsed host : %.*s", hostlen, host->av_val);
99.         } else {
100.             RTMP_Log(RTMP_LOGWARNING, "Hostname exceeds 255 characters!");
101.         }
102.
103.         p+=hostlen;
104.     }
105.
106.     /* 获取端口号 */

```

```

107.     if(*p == ':') {
108.         unsigned int p2;
109.         p++;
110.         p2 = atoi(p);
111.         if(p2 > 65535) {
112.             RTMP_Log(RTMP_LOGWARNING, "Invalid port number!");
113.         } else {
114.             *port = p2;
115.         }
116.     }
117.
118.     if(!slash) {
119.         RTMP_Log(RTMP_LOGWARNING, "No application or playpath in URL!");
120.         return TRUE;
121.     }
122.     p = slash+1;
123.
124.     {
125.         /* 获取应用程序(application)
126.          *
127.          * rtmp://host[:port]/app[/appinstance][...]
128.          * application = app[/appinstance]
129.          */
130.
131.         char *slash2, *slash3 = NULL; //指向第二个斜杠, 第三个斜杠的指针
132.         int applen, appnamelen;
133.
134.         slash2 = strchr(p, '/'); //指向第二个斜杠
135.         if(slash2)
136.             slash3 = strchr(slash2+1, '/'); //指向第三个斜杠, 注意slash2之所以+1是因为让其右移一位
137.
138.         applen = end-p; /* ondemand, pass all parameters as app */
139.         appnamelen = applen; /* ondemand length */
140.
141.         if(ques && strstr(p, "slist=")) { /* whatever it is, the '?' and slist= means we need to use everything as app and parse plapath
142.             om slist= */
143.             appnamelen = ques-p;
144.         }
145.         else if(strncmp(p, "ondemand/", 9)==0) {
146.             /* app = ondemand/foobar, only pass app=ondemand */
147.             applen = 8;
148.             appnamelen = 8;
149.         }
150.         else { /* app!=ondemand, so app is app[/appinstance] */
151.             if(slash3)
152.                 appnamelen = slash3-p;
153.             else if(slash2)
154.                 appnamelen = slash2-p;
155.
156.             applen = appnamelen;
157.         }
158.
159.         app->av_val = p;
160.         app->av_len = applen;
161.         RTMP_Log(RTMP_LOGDEBUG, "Parsed app      : %.*s", applen, p);
162.
163.         p += appnamelen;
164.     }
165.
166.     if (*p == '/')
167.         p++;
168.
169.     if (end-p) {
170.         AVal av = {p, end-p};
171.         RTMP_ParsePlaypath(&av, playpath);
172.     }
173.
174.     return TRUE;
175. }
176.
177. /*
178.  * 从URL中获取播放路径 (playpath)。播放路径是URL中“rtmp://host:port/app/”后面的部分
179.  *
180.  * 获取FMS能够识别的播放路径
181.  * mp4 流: 前面添加 "mp4:", 删除扩展名
182.  * mp3 流: 前面添加 "mp3:", 删除扩展名
183.  * flv 流: 删除扩展名
184.  */
185. void RTMP_ParsePlaypath(AVal *in, AVal *out) {
186.     int addMP4 = 0;
187.     int addMP3 = 0;
188.     int subExt = 0;
189.     const char *playpath = in->av_val;
190.     const char *temp, *q, *ext = NULL;
191.     const char *ppstart = playpath;
192.     char *streamname, *destptr, *p;
193.
194.     int pplen = in->av_len;
195.
196.     out->av_val = NULL;
197.     out->av_len = 0;

```

```

198.     if ((*ppstart == '?') &&
199.         (temp=strstr(ppstart, "slist=")) != 0) {
200.         ppstart = temp+6;
201.         pplen = strlen(ppstart);
202.
203.         temp = strchr(ppstart, '&');
204.         if (temp) {
205.             pplen = temp-ppstart;
206.         }
207.     }
208.
209.     q = strchr(ppstart, '?');
210.     if (pplen >= 4) {
211.         if (q)
212.             ext = q-4;
213.         else
214.             ext = &ppstart[pplen-4];
215.         if ((strncmp(ext, ".f4v", 4) == 0) ||
216.             (strncmp(ext, ".mp4", 4) == 0)) {
217.             addMP4 = 1;
218.             subExt = 1;
219.             /* Only remove .flv from rtmp URL, not slist params */
220.         } else if ((ppstart == playpath) &&
221.                    (strncmp(ext, ".flv", 4) == 0)) {
222.             subExt = 1;
223.         } else if (strncmp(ext, ".mp3", 4) == 0) {
224.             addMP3 = 1;
225.             subExt = 1;
226.         }
227.     }
228.
229.     streamname = (char *)malloc((pplen+4+1)*sizeof(char));
230.     if (!streamname)
231.         return;
232.
233.     destptr = streamname;
234.     if (addMP4) {
235.         if (strncmp(ppstart, "mp4:", 4)) {
236.             strcpy(destptr, "mp4:");
237.             destptr += 4;
238.         } else {
239.             subExt = 0;
240.         }
241.     } else if (addMP3) {
242.         if (strncmp(ppstart, "mp3:", 4)) {
243.             strcpy(destptr, "mp3:");
244.             destptr += 4;
245.         } else {
246.             subExt = 0;
247.         }
248.     }
249.
250.     for (p=(char *)ppstart; pplen >0;) {
251.         /* skip extension */
252.         if (subExt && p == ext) {
253.             p += 4;
254.             pplen -= 4;
255.             continue;
256.         }
257.         if (*p == '%') {
258.             unsigned int c;
259.             sscanf(p+1, "%02x", &c);
260.             *destptr++ = c;
261.             pplen -= 3;
262.             p += 3;
263.         } else {
264.             *destptr++ = *p++;
265.             pplen--;
266.         }
267.     }
268.     *destptr = '\0';
269.
270.     out->av_val = streamname;
271.     out->av_len = destptr - streamname;
272. }

```

rtmpdump源代码 (Linux) : <http://download.csdn.net/detail/leixiaohua1020/6376561>

rtmpdump源代码 (VC 2005 工程) : <http://download.csdn.net/detail/leixiaohua1020/6563163>

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/12953833>

文章标签： [RTMPDump](#) [源代码](#) [RTMP\\_ParseURL](#) [url](#) [解析](#)

个人分类： [libRTMP](#)

此PDF由[spygg](#)生成,请尊重原作者版权!!!  
我的邮箱:liushidc@163.com