

原 live555 源代码简单分析1：主程序

2013年09月25日 17:36:24 阅读数：12906

live555是使用十分广泛的开源流媒体服务器，之前也看过其他人写的live555的学习笔记，在这里自己简单总结下。

live555源代码有以下几个明显的特点：

1.头文件是.hh后缀的，但没觉得和.h后缀的有什么不同

2.采用了面向对象的程序设计思路，里面各种对象

好了，不罗嗦，使用vc2010打开live555的vc工程，看到live555源代码结构如下：

□

源代码由5个工程构成（4个库和一个主程序）：

libUsageEnvironment.lib；libliveMedia.lib；libgroupsock.lib；libBasicUsageEnvironment.lib；以及live555MediaServer

这里我们只分析live555MediaServer这个主程序，其实代码量并不大，主要有两个CPP：DynamicRTSPServer.cpp和live555MediaServer.cpp

程序的main()在live555MediaServer.cpp中，在main()中调用了DynamicRTSPServer中的类

不废话，直接贴上有注释的源码

live555MediaServer.cpp：

```

1. #include <BasicUsageEnvironment.hh>
2. #include "DynamicRTSPServer.hh"
3. #include "version.hh"
4.
5. int main(int argc, char** argv) {
6.     // Begin by setting up our usage environment:
7.     // TaskScheduler用于任务计划
8.     TaskScheduler* scheduler = BasicTaskScheduler::createNew();
9.     // UsageEnvironment用于输出
10.    UsageEnvironment* env = BasicUsageEnvironment::createNew(*scheduler);
11.
12.    UserAuthenticationDatabase* authDB = NULL;
13.    #ifdef ACCESS_CONTROL
14.        // To implement client access control to the RTSP server, do the following:
15.        authDB = new UserAuthenticationDatabase;
16.        authDB->addUserRecord("username1", "password1"); // replace these with real strings
17.        // Repeat the above with each <username>, <password> that you wish to allow
18.        // access to the server.
19.    #endif
20.
21.    //建立 RTSP server. 使用默认端口 (554),
22.    // and then with the alternative port number (8554):
23.    RTSPServer* rtspServer;
24.    portNumBits rtspServerPortNum = 554;
25.    //创建 RTSPServer实例
26.    rtspServer = DynamicRTSPServer::createNew(*env, rtspServerPortNum, authDB);
27.    if (rtspServer == NULL) {
28.        rtspServerPortNum = 8554;
29.        rtspServer = DynamicRTSPServer::createNew(*env, rtspServerPortNum, authDB);
30.    }
31.    if (rtspServer == NULL) {
32.        *env << "Failed to create RTSP server: " << env->getResultMsg() << "\n";
33.        exit(1);
34.    }
35.    //用到了运算符重载
36.    *env << "LIVE555 Media Server\n";
37.    *env << "\tversion " << MEDIA_SERVER_VERSION_STRING
38.        << " (LIVE555 Streaming Media Library version "
39.        << LIVEMEDIA_LIBRARY_VERSION_STRING << ").\n";
40.
41.    char* urlPrefix = rtspServer->rtspURLPrefix();
42.    *env << "Play streams from this server using the URL\n\t"
43.        << urlPrefix << "<filename>\nwhere <filename> is a file present in the current directory.\n";
44.    *env << "Each file's type is inferred from its name suffix:\n";
45.    *env << "\t\".aac\" => an AAC Audio (ADTS format) file\n";
46.    *env << "\t\".amr\" => an AMR Audio file\n";
47.    *env << "\t\".m4e\" => a MPEG-4 Video Elementary Stream file\n";
48.    *env << "\t\".dv\" => a DV Video file\n";
49.    *env << "\t\".mp3\" => a MPEG-1 or 2 Audio file\n";
50.    *env << "\t\".mpg\" => a MPEG-1 or 2 Program Stream (audio+video) file\n";
51.    *env << "\t\".ts\" => a MPEG Transport Stream file\n";
52.    *env << "\t\t(a \".tsx\" index file - if present - provides server 'trick play' support)\n";
53.    *env << "\t\".wav\" => a WAV Audio file\n";
54.    *env << "See http://www.live555.com/mediaServer/ for additional documentation.\n";
55.
56.    // Also, attempt to create a HTTP server for RTSP-over-HTTP tunneling.
57.    // Try first with the default HTTP port (80), and then with the alternative HTTP
58.    // port numbers (8000 and 8080).
59.
60.    if (rtspServer->setUpTunnelingOverHTTP(80) || rtspServer->setUpTunnelingOverHTTP(8000) || rtspServer-
61.        >setUpTunnelingOverHTTP(8080)) {
62.        *env << "(We use port " << rtspServer->httpServerPortNum() << " for optional RTSP-over-HTTP tunneling.)\n";
63.    } else {
64.        *env << "(RTSP-over-HTTP tunneling is not available.)\n";
65.    }
66.    //进入一个永久的循环
67.    env->taskScheduler().doEventLoop(); // does not return
68.
69.    return 0; // only to prevent compiler warning
70. }

```

DynamicRTSPServer.cpp :

```

1. #include "DynamicRTSPServer.hh"
2. #include <liveMedia.hh>
3. #include <string.h>
4.
5. DynamicRTSPServer*
6. DynamicRTSPServer::createNew(UsageEnvironment& env, Port ourPort,
7.    UserAuthenticationDatabase* authDatabase,
8.    unsigned reclamationTestSeconds) {
9.    int ourSocket = -1;
10.
11.    do {
12.        //建立TCP socket(socket(),bind(),listen()...)
13.        int ourSocket = setUpOurSocket(env, ourPort);

```

```

14.     if (ourSocket == -1) break;
15.
16.     return new DynamicRTSPServer(env, ourSocket, ourPort, authDatabase, reclamationTestSeconds);
17. } while (0);
18.
19. if (ourSocket != -1) ::closeSocket(ourSocket);
20. return NULL;
21. }
22.
23. DynamicRTSPServer::DynamicRTSPServer(UsageEnvironment& env, int ourSocket,
24.                                     Port ourPort,
25.                                     UserAuthenticationDatabase* authDatabase, unsigned reclamationTestSeconds)
26. : RTSPServer(env, ourSocket, ourPort, authDatabase, reclamationTestSeconds) {
27. }
28.
29. DynamicRTSPServer::~DynamicRTSPServer() {
30. }
31.
32. static ServerMediaSession* createNewSMS(UsageEnvironment& env,
33.                                     char const* fileName, FILE* fid); // forward
34.
35.
36.
37. //查找ServerMediaSession (对应服务器上一个媒体文件, , 或设备), 如果没有的话就创建一个
38. //streamName例:A.avi
39. ServerMediaSession*
40. DynamicRTSPServer::lookupServerMediaSession(char const* streamName) {
41.     // First, check whether the specified "streamName" exists as a local file:
42.     FILE* fid = fopen(streamName, "rb");
43.     //如果返回文件指针不为空, 则文件存在
44.     Boolean fileExists = fid != NULL;
45.
46.     // Next, check whether we already have a "ServerMediaSession" for this file:
47.     //看看是否有这个ServerMediaSession
48.     ServerMediaSession* sms = RTSPServer::lookupServerMediaSession(streamName);
49.     Boolean smsExists = sms != NULL;
50.
51.     // Handle the four possibilities for "fileExists" and "smsExists":
52.     //文件没了, ServerMediaSession有, 删之
53.     if (!fileExists) {
54.         if (smsExists) {
55.             // "sms" was created for a file that no longer exists. Remove it:
56.             removeServerMediaSession(sms);
57.         }
58.         return NULL;
59.     } else {
60.         //文件有, ServerMediaSession无, 加之
61.         if (!smsExists) {
62.             // Create a new "ServerMediaSession" object for streaming from the named file.
63.             sms = createNewSMS(envir(), streamName, fid);
64.             addServerMediaSession(sms);
65.         }
66.         fclose(fid);
67.         return sms;
68.     }
69. }
70.
71. #define NEW_SMS(description) do {\
72. char const* descStr = description\
73.     ", streamed by the LIVE555 Media Server";\
74. sms = ServerMediaSession::createNew(env, fileName, descStr);\
75. } while(0)
76.
77.
78. //创建一个ServerMediaSession
79. static ServerMediaSession* createNewSMS(UsageEnvironment& env,
80.                                     char const* fileName, FILE* /*fid*/) {
81.     // Use the file name extension to determine the type of "ServerMediaSession":
82.     //获取扩展名, 以"."开始. 不严密, 万一文件名有多个点?
83.     char const* extension = strrchr(fileName, '.');
84.     if (extension == NULL) return NULL;
85.
86.     ServerMediaSession* sms = NULL;
87.     Boolean const reuseSource = False;
88.     if (strcmp(extension, ".aac") == 0) {
89.         // Assumed to be an AAC Audio (ADTS format) file:
90.         // 调用ServerMediaSession::createNew ()
91.         //还会调用MediaSubsession
92.         NEW_SMS("AAC Audio");
93.         sms->addSubsession(ADTSAudioFileServerMediaSubsession::createNew(env, fileName, reuseSource));
94.     } else if (strcmp(extension, ".amr") == 0) {
95.         // Assumed to be an AMR Audio file:
96.         NEW_SMS("AMR Audio");
97.         sms->addSubsession(AMRAudioFileServerMediaSubsession::createNew(env, fileName, reuseSource));
98.     } else if (strcmp(extension, ".m4e") == 0) {
99.         // Assumed to be a MPEG-4 Video Elementary Stream file:
100.        NEW_SMS("MPEG-4 Video");
101.        sms->addSubsession(MPEG4VideoFileServerMediaSubsession::createNew(env, fileName, reuseSource));
102.    } else if (strcmp(extension, ".mp3") == 0) {
103.        // Assumed to be a MPEG-1 or 2 Audio file:
104.        NEW_SMS("MPEG-1 or 2 Audio");

```

```

105. // To stream using 'ADUs' rather than raw MP3 frames, uncomment the following:
106. // #define STREAM_USING_ADUS 1
107. // To also reorder ADUs before streaming, uncomment the following:
108. // #define INTERLEAVE_ADUS 1
109. // (For more information about ADUs and interleaving,
110. // see <http://www.live555.com/rtp-mp3/>)
111. Boolean useADUs = False;
112. Interleaving* interleaving = NULL;
113. #ifndef STREAM_USING_ADUS
114. useADUs = True;
115. #ifndef INTERLEAVE_ADUS
116. unsigned char interleaveCycle[] = {0,2,1,3}; // or choose your own...
117. unsigned const interleaveCycleSize
118. = (sizeof interleaveCycle)/(sizeof unsigned char);
119. interleaving = new Interleaving(interleaveCycleSize, interleaveCycle);
120. #endif
121. #endif
122. sms->addSubsession(MP3AudioFileServerMediaSubsession::createNew(env, fileName, reuseSource, useADUs, interleaving));
123. } else if (strcmp(extension, ".mpg") == 0) {
124. // Assumed to be a MPEG-1 or 2 Program Stream (audio+video) file:
125. NEW_SMS("MPEG-1 or 2 Program Stream");
126. MPEG1or2FileServerDemux* demux
127. = MPEG1or2FileServerDemux::createNew(env, fileName, reuseSource);
128. sms->addSubsession(demux->newVideoServerMediaSubsession());
129. sms->addSubsession(demux->newAudioServerMediaSubsession());
130. } else if (strcmp(extension, ".ts") == 0) {
131. // Assumed to be a MPEG Transport Stream file:
132. // Use an index file name that's the same as the TS file name, except with ".tsx":
133. unsigned indexFileNameLen = strlen(fileName) + 2; // allow for trailing "\0"
134. char* indexFileName = new char[indexFileNameLen];
135. sprintf(indexFileName, "%sx", fileName);
136. NEW_SMS("MPEG Transport Stream");
137. sms->addSubsession(MPEG2TransportFileServerMediaSubsession::createNew(env, fileName, indexFileName, reuseSource));
138. delete[] indexFileName;
139. } else if (strcmp(extension, ".wav") == 0) {
140. // Assumed to be a WAV Audio file:
141. NEW_SMS("WAV Audio Stream");
142. // To convert 16-bit PCM data to 8-bit u-law, prior to streaming,
143. // change the following to True:
144. Boolean convertToULaw = False;
145. sms->addSubsession(WAVAudioFileServerMediaSubsession::createNew(env, fileName, reuseSource, convertToULaw));
146. } else if (strcmp(extension, ".dv") == 0) {
147. // Assumed to be a DV Video file
148. // First, make sure that the RTPSinks' buffers will be large enough to handle the huge size of DV frames (as big as 288000).
149. OutPacketBuffer::maxSize = 300000;
150.
151. NEW_SMS("DV Video");
152. sms->addSubsession(DVVideoFileServerMediaSubsession::createNew(env, fileName, reuseSource));
153. }
154.
155. return sms;
156. }

```

live555源代码（VC6）：<http://download.csdn.net/detail/leixiaohua1020/6374387>

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/12022409>

文章标签：[live555](#) [源代码](#) [流媒体服务器](#) [分析](#) [主程序](#)

个人分类：[Live555](#)

所属专栏：[开源多媒体项目源代码分析](#)

此PDF由spyyg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com