原

live555 源代码简单分析1：主程序

2013年09月25日 17:36:24

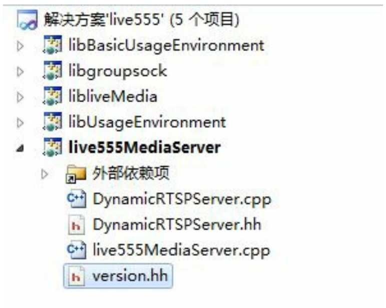
live555是使用十分广泛的开源流媒体服务器，之前也看过其他人写的live555的学习笔记，在这里自己简单总结下。

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：

```cpp
#include <BasicUsageEnvironment.hh>
#include "DynamicRTSPServer.hh"
#include "version.hh"

int main(int argc, char** argv) {
  // Begin by setting up our usage environment:
  // TaskScheduler用于任务计划
  TaskScheduler* scheduler = BasicTaskScheduler::createNew();
  // UsageEnvironment用于输出
  UsageEnvironment* env = BasicUsageEnvironment::createNew(*scheduler);

  UserAuthenticationDatabase* authDB = NULL;
#ifdef ACCESS_CONTROL
  // To implement client access control to the RTSP server, do the following:
  authDB = new UserAuthenticationDatabase;
  authDB->addUserRecord("username1", "password1"); // replace these with real strings
  // Repeat the above with each <username>, <password> that you wish to allow
  // access to the server.
#endif

  //建立 RTSP server.  使用默认端口 (554),
  // and then with the alternative port number (8554):
  RTSPServer* rtspServer;
  portNumBits rtspServerPortNum = 554;
  //创建 RTSPServer实例
  rtspServer = DynamicRTSPServer::createNew(*env, rtspServerPortNum, authDB);
  if (rtspServer == NULL) {
    rtspServerPortNum = 8554;
    rtspServer = DynamicRTSPServer::createNew(*env, rtspServerPortNum, authDB);
  }
  if (rtspServer == NULL) {
    *env << "Failed to create RTSP server: " << env->getResultMsg() << "\n";
    exit(1);
  }
  //用到了运算符重载
  *env << "LIVE555 Media Server\n";
  *env << "\tversion " << MEDIA_SERVER_VERSION_STRING
       << " (LIVE555 Streaming Media library version "
       << LIVEMEDIA_LIBRARY_VERSION_STRING << ").\n";

  char* urlPrefix = rtspServer->rtspURLPrefix();
  *env << "Play streams from this server using the URL\n\t"
       << urlPrefix << "<filename>\nwhere <filename> is a file present in the current directory.\n";
  *env << "Each file's type is inferred from its name suffix:\n";
  *env << "\t\".aac\" => an AAC Audio (ADTS format) file\n";
  *env << "\t\".amr\" => an AMR Audio file\n";
  *env << "\t\".m4e\" => a MPEG-4 Video Elementary Stream file\n";
  *env << "\t\".dv\" => a DV Video file\n";
  *env << "\t\".mp3\" => a MPEG-1 or 2 Audio file\n";
  *env << "\t\".mpg\" => a MPEG-1 or 2 Program Stream (audio+video) file\n";
  *env << "\t\".ts\" => a MPEG Transport Stream file\n";
  *env << "\t\t(a \".tsx\" index file - if present - provides server 'trick play' support)\n";
  *env << "\t\".wav\" => a WAV Audio file\n";
  *env << "See http://www.live555.com/mediaServer/ for additional documentation.\n";

  // Also, attempt to create a HTTP server for RTSP-over-HTTP tunneling.
  // Try first with the default HTTP port (80), and then with the alternative HTTP
  // port numbers (8000 and 8080).

  if (rtspServer->setUpTunnelingOverHTTP(80) || rtspServer->setUpTunnelingOverHTTP(8000) || rtspServer->setUpTunnelingOverHTTP(8080)) {
    *env << "(We use port " << rtspServer->httpServerPortNum() << " for optional RTSP-over-HTTP tunneling.)\n";
  } else {
    *env << "(RTSP-over-HTTP tunneling is not available.)\n";
  }
  //进入一个永久的循环
  env->taskScheduler().doEventLoop(); // does not return

  return 0; // only to prevent compiler warning
}
```

DynamicRTSPServer.cpp：

```cpp
#include "DynamicRTSPServer.hh"
#include <liveMedia.hh>
#include <string.h>

DynamicRTSPServer*
DynamicRTSPServer::createNew(UsageEnvironment& env, Port ourPort,
                 UserAuthenticationDatabase* authDatabase,
                 unsigned reclamationTestSeconds) {
  int ourSocket = -1;

  do {
    //建立TCP socket(socket(),bind(),listen()...)
    int ourSocket = setUpOurSocket(env, ourPort);
```

```
    if (ourSocket == -1) break;

    return new DynamicRTSPServer(env, ourSocket, ourPort, authDatabase, reclamationTestSeconds);
  } while (0);

  if (ourSocket != -1) ::closeSocket(ourSocket);
  return NULL;
}

DynamicRTSPServer::DynamicRTSPServer(UsageEnvironment& env, int ourSocket,
                     Port ourPort,
                     UserAuthenticationDatabase* authDatabase, unsigned reclamationTestSeconds)
  : RTSPServer(env, ourSocket, ourPort, authDatabase, reclamationTestSeconds) {
}

DynamicRTSPServer::~DynamicRTSPServer() {
}

static ServerMediaSession* createNewSMS(UsageEnvironment& env,
                    char const* fileName, FILE* fid); // forward



//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
//streamName例:A.avi
ServerMediaSession*
DynamicRTSPServer::lookupServerMediaSession(char const* streamName) {
  // First, check whether the specified "streamName" exists as a local file:
  FILE* fid = fopen(streamName, "rb");
  //如果返回文件指针不为空，则文件存在
  Boolean fileExists = fid != NULL;

  // Next, check whether we already have a "ServerMediaSession" for this file:
  //看看是否有这个ServerMediaSession
  ServerMediaSession* sms = RTSPServer::lookupServerMediaSession(streamName);
  Boolean smsExists = sms != NULL;

  // Handle the four possibilities for "fileExists" and "smsExists":
  //文件没了，ServerMediaSession有，删之
  if (!fileExists) {
    if (smsExists) {
      // "sms" was created for a file that no longer exists. Remove it:
      removeServerMediaSession(sms);
    }
    return NULL;
  } else {
    //文件有，ServerMediaSession无，加之
    if (!smsExists) {
      // Create a new "ServerMediaSession" object for streaming from the named file.
      sms = createNewSMS(envir(), streamName, fid);
      addServerMediaSession(sms);
    }
    fclose(fid);
    return sms;
  }
}

#define NEW_SMS(description) do {\
char const* descStr = description\
    ", streamed by the LIVE555 Media Server";\
sms = ServerMediaSession::createNew(env, fileName, fileName, descStr);\
} while(0)


//创建一个ServerMediaSession
static ServerMediaSession* createNewSMS(UsageEnvironment& env,
                    char const* fileName, FILE* /*fid*/) {
  // Use the file name extension to determine the type of "ServerMediaSession":
    //获取扩展名，以“.”开始。不严密，万一文件名有多个点？
  char const* extension = strrchr(fileName, '.');
  if (extension == NULL) return NULL;

  ServerMediaSession* sms = NULL;
  Boolean const reuseSource = False;
  if (strcmp(extension, ".aac") == 0) {
    // Assumed to be an AAC Audio (ADTS format) file:
    // 调用ServerMediaSession::createNew（）
    //还会调用MediaSubsession
    NEW_SMS("AAC Audio");
    sms->addSubsession(ADTSAudioFileServerMediaSubsession::createNew(env, fileName, reuseSource));
  } else if (strcmp(extension, ".amr") == 0) {
    // Assumed to be an AMR Audio file:
    NEW_SMS("AMR Audio");
    sms->addSubsession(AMRAudioFileServerMediaSubsession::createNew(env, fileName, reuseSource));
  } else if (strcmp(extension, ".m4e") == 0) {
    // Assumed to be a MPEG-4 Video Elementary Stream file:
    NEW_SMS("MPEG-4 Video");
    sms->addSubsession(MPEG4VideoFileServerMediaSubsession::createNew(env, fileName, reuseSource));
  } else if (strcmp(extension, ".mp3") == 0) {
    // Assumed to be a MPEG-1 or 2 Audio file:
    NEW_SMS("MPEG-1 or 2 Audio");
```

```
    // To stream using 'ADUs' rather than raw MP3 frames, uncomment the following:
//#define STREAM_USING_ADUS 1
    // To also reorder ADUs before streaming, uncomment the following:
//#define INTERLEAVE_ADUS 1
    // (For more information about ADUs and interleaving,
    //  see <http://www.live555.com/rtp-mp3/>)
    Boolean useADUs = False;
    Interleaving* interleaving = NULL;
#ifdef STREAM_USING_ADUS
    useADUs = True;
#ifdef INTERLEAVE_ADUS
    unsigned char interleaveCycle[] = {0,2,1,3}; // or choose your own...
    unsigned const interleaveCycleSize
      = (sizeof interleaveCycle)/(sizeof (unsigned char));
    interleaving = new Interleaving(interleaveCycleSize, interleaveCycle);
#endif
#endif
    sms->addSubsession(MP3AudioFileServerMediaSubsession::createNew(env, fileName, reuseSource, useADUs, interleaving));
  } else if (strcmp(extension, ".mpg") == 0) {
    // Assumed to be a MPEG-1 or 2 Program Stream (audio+video) file:
    NEW_SMS("MPEG-1 or 2 Program Stream");
    MPEG1or2FileServerDemux* demux
      = MPEG1or2FileServerDemux::createNew(env, fileName, reuseSource);
    sms->addSubsession(demux->newVideoServerMediaSubsession());
    sms->addSubsession(demux->newAudioServerMediaSubsession());
  } else if (strcmp(extension, ".ts") == 0) {
    // Assumed to be a MPEG Transport Stream file:
    // Use an index file name that's the same as the TS file name, except with ".tsx":
    unsigned indexFileNameLen = strlen(fileName) + 2; // allow for trailing "x\0"
    char* indexFileName = new char[indexFileNameLen];
    sprintf(indexFileName, "%sx", fileName);
    NEW_SMS("MPEG Transport Stream");
    sms->addSubsession(MPEG2TransportFileServerMediaSubsession::createNew(env, fileName, indexFileName, reuseSource));
    delete[] indexFileName;
  } else if (strcmp(extension, ".wav") == 0) {
    // Assumed to be a WAV Audio file:
    NEW_SMS("WAV Audio Stream");
    // To convert 16-bit PCM data to 8-bit u-law, prior to streaming,
    // change the following to True:
    Boolean convertToULaw = False;
    sms->addSubsession(WAVAudioFileServerMediaSubsession::createNew(env, fileName, reuseSource, convertToULaw));
  } else if (strcmp(extension, ".dv") == 0) {
    // Assumed to be a DV Video file
    // First, make sure that the RTPSinks' buffers will be large enough to handle the huge size of DV frames (as big as 288000).
    OutPacketBuffer::maxSize = 300000;

    NEW_SMS("DV Video");
    sms->addSubsession(DVVideoFileServerMediaSubsession::createNew(env, fileName, reuseSource));
  }

  return sms;
}
```

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

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