# 服务器

## 目录结构与运行

源码：/sample/asr/src 下面

运行：/sample/asr/run 中，

执行：make，产生可执行文件asr和一个空的文件夹wav

运行：./asr

wav文件夹用于存放客户端传来的音频文件

**请确保所有文件权限属于普通用户，否则可能会创建音频文件会失败**

可运行环境：Ubuntu16.04LTS，Ubuntu14.04LTS

如要更换IP和端口，请打开main.cpp修改代码

## 源码解析

### main.cpp:

主要流程：

1. 创建Tcp连接
2. 创建客户线程
3. 接收文件
4. 翻译音频文件，并发送结果

### TcpSocket.cpp和TcpSocket.h:

创建Tcp连接，并接受客户端

### FileHandle.cpp和FileHandle.h:

创建并且接受音频文件。

#### 传输数据帧

注意：客户端发送的文件帧格式：

|  |  |  |
| --- | --- | --- |
| 8字节 | 8字节 | 文件名 |

文件名+文件大小 文件名大小

int64 Str2Int(char \*str)函数：

将字符串转换成正确的长整形，可以计算出总大小和文件名大小

string GetFileName(char \*filename, int64 size)：

因为客户端发送帧采用QByteArray，所以每个字符会占两个字节，若文件名遵守ASCII，则会由一个字节是空，服务器接收到之后，该函数会忽略空字符，从而接收到正确的文件名，所以**请保证文件名遵守ASCII**

### Translate.cpp和Translate.h：

翻译音频文件

### 其他文件均为例2的文件

# 客户端

## 目录结构与运行

源码：/code 中

运行：Client.exe

wav文件夹：存放临时的录音文件

界面：

主界面：网络操作，文件操作

录音界面：录制音频文件

## 源码解析

### mainwindow.cpp和mainwindow.h

将网络界面作为主界面，录音界面通过点击菜单上的按钮来弹出

### netwidget.cpp和netwidget.h

QString GetResult(); //展示服务器发来的结果

void CloseConnect(); //界面关闭时，关闭连接

void showError(QAbstractSocket::SocketError); //展示错误

void SendFileName(); //发送文件名

void SendFileData(qint64); //发送文件

void RecvMessage(); //接受服务器发来的结果

void GetRecord(QString fileName); //录音结束，自动发送该音频文件

接收到服务器的音频解析结果为Json格式：

{

"results\_recognition":["喂喂喂","喂喂喂喂","喂喂卫","喂喂喂为","为为为","喂喂为","为喂喂"],

"origin\_result":

{

"corpus\_no":6550156045544704337,

"err\_no":0,

"result":

{

"word":["喂喂喂","喂喂喂喂","喂喂卫","喂喂喂为","为为为","喂喂为","为喂喂"]

},

"sn":"F1185C1E-DFFE-4078-B40D-4CB5B00C51DE",

"voice\_energy":9806.56933593750

},

"sn\_start\_time":"00:01.200",

"sn\_end\_time":"00:03.560"

}

所以结果取results\_recognition中前两个结果，进行展示

### recordwidget.cpp和recordwidget.h

录音

void RecordOk(QString fileName); //录音结束发送该信号，通知netwidget自动发送音频文件

# 交互流程