

一、文件架构

二、分析SDK的例程

三、如何使用SDK原生包

四、运行演示

1.运行步骤

2.语音引擎工作过程

一、文件架构

```
x86/
    bin/
        msc/  ---> 识别的记录文件
        pcm/  ---> 识别的音频文件
        asr_record_demo  ---> 可执行文件[*]
        cmd.bnf  ---> 识别的语法文件[*]
        result.xml: SDK识别的结果存档
    example/
        asr_record_demo: SDK的例程
    lib/
        SDK的识别库
    BNF语法手册.pdf
```

Makefile: 工程管理文件，用于编译程序

二、分析SDK的例程

例程源码: x86/examples/asr_record_demo/asr_record_demo.c

2.1 登录账号

```
MSPLogin();
```

2.2 构建语法网络

```
build_grammar();  ---> 加载x86/bin/cmd.bnf语法  ---> 需要一定的时间
```

2.3 开始识别

```
run_asr();
```

2.3.1 与开发板端建立连接

```
init_sock();
socket();  ---> 建立未连接套接字
bind();  ---> 绑定IP地址作为服务器 192.168.0.68
listen();  ---> 把未连接套接字设置为监听套接字
accept();  ---> 接受对方的连接  ---> 得到已连接套接字
```

2.3.2 获取音频文件

```
get_audio_file();  --> return "pcm/cmd.pcm"
```

2.3.3 分析音频文件

```
demo_file("pcm/cmd.pcm");
```

```
struct speech_rec_notifier renotifier =
{
    on_result,  ---->识别的结果
```

```

        on_speech_begin, ---->开始下一次的识别
        on_speech_end ----> 识别结束
    };

```

2.3.4 初始化引擎语法

```
sr_init();
```

2.3.5 把音频传入引擎中，进行识别

```
sr_start_listening();
```

2.3.6 不断进行识别 ---> 引擎不是一下子就识别完毕

```
sr_write_audio_data();
```

2.3.7 识别完毕，把结果存在on_result

```
sr_stop_listening();
```

```
on_result() ---> show_result() ---> write(开发板的套接字)
```

三、如何使用SDK原生包

3.1 修改语法

```
/x86/bin/cmd.bnf
```

3.2 编译SDK

```
cd ~/x86/examples/asr_record_demo
make
```

3.3 切换可执行文件的路径下

```
cd ~/x86/bin
./asr_record_demo
构建离线识别语法网络...
构建语法失败! 11212 ----> 当前的时间不在SDK的试用日期内
语音识别完毕.
```

3.4 修改Ubuntu日期为2016.12.15

```
date -s 12/15/2016
```

3.5 再次执行文件

```
./asr_record_demo
构建离线识别语法网络...
构建语法成功! 语法ID:cmd
离线识别语法网络构建完成，开始识别...
wait for connecting ...
```

3.6 如果想增加更加多的语音识别，必须去修改/x86/bin/cmd.bnf, 例如增加小爱同学的语音，示例如下：

```
<cmd>:开窗!id(1)|开门!id(2)|开灯!id(3)|关窗!id(4)|关门!id(5)|关灯!id(6)|小爱同学!id(7)|小飞同学!id(7)|退出!id(999)|不玩了!id(999);
```

添加完毕之后，必须重新编译该工程。

```

1 #x86/examples/asr_record_demo
2 #make clean
3 #make

```

注：

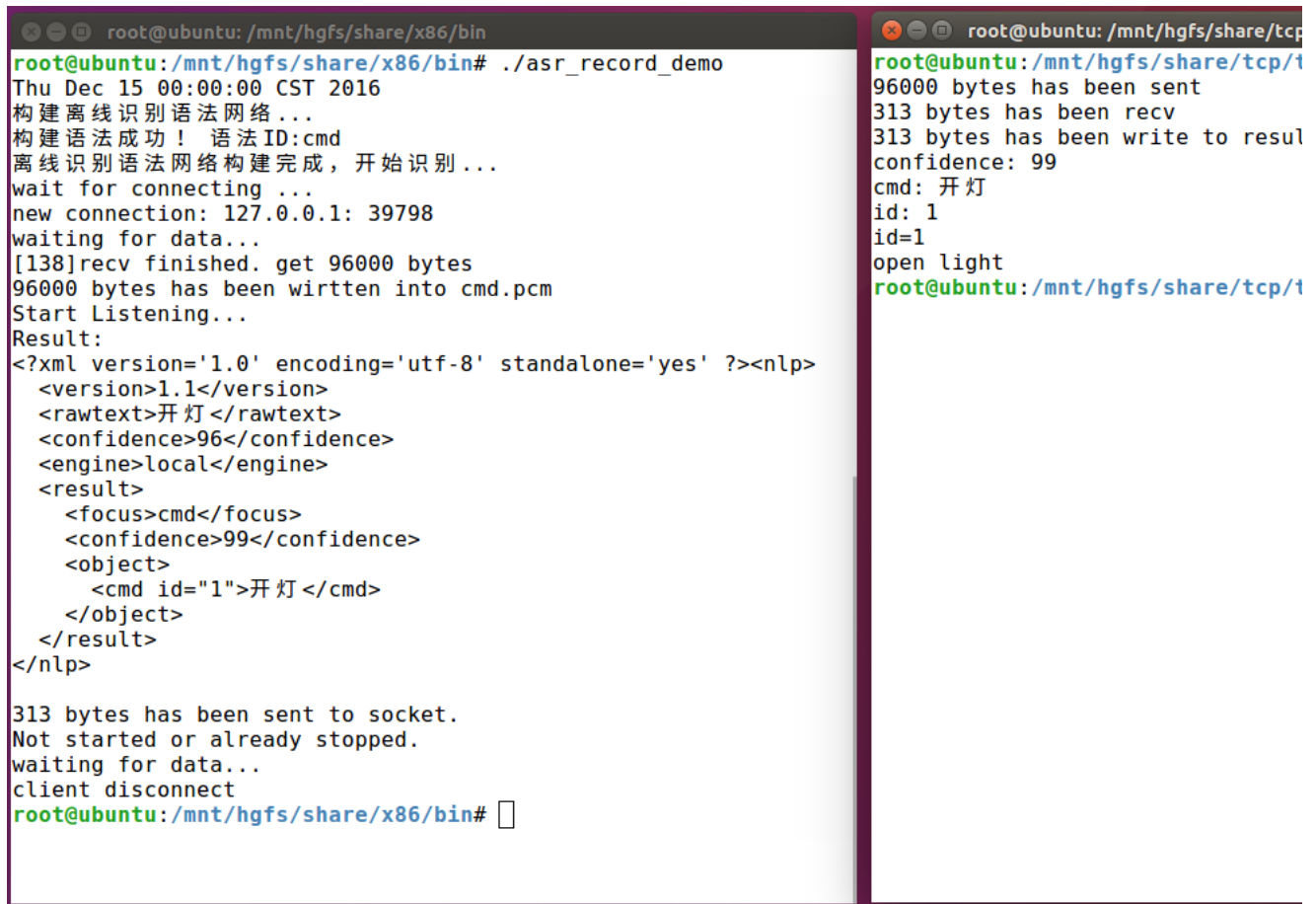
修改cmd.bnf文件的时候，最好将该文件拷贝共享目录，在windows平台进行修改，因为windows对中文支持非常友好。修改后，再重新拷贝覆盖原文件。

四、运行演示

1. 运行步骤

- (1) 运行科大讯飞语音引擎中的asr_record_demo程序。
- (2) 运行温老师提供的tcp_xml_test程序。

以下为运行结果图。



The image shows two terminal windows side-by-side. The left window is titled 'root@ubuntu: /mnt/hgfs/share/x86/bin' and shows the execution of the 'asr_record_demo' program. It displays the process of building an offline recognition grammar network, waiting for a connection, receiving 96000 bytes of audio data, and then outputting an XML result with a confidence of 99 for the command '开灯' (turn on the light). The right window is titled 'root@ubuntu: /mnt/hgfs/share/tcp/1' and shows the execution of the 'tcp_xml_test' program, which displays the received data and the command '开灯'.

```
root@ubuntu: /mnt/hgfs/share/x86/bin# ./asr_record_demo
Thu Dec 15 00:00:00 CST 2016
构建离线识别语法网络...
构建语法成功! 语法ID:cmd
离线识别语法网络构建完成, 开始识别...
wait for connecting ...
new connection: 127.0.0.1: 39798
waiting for data...
[138]recv finished. get 96000 bytes
96000 bytes has been wirtten into cmd.pcm
Start Listening...
Result:
<?xml version='1.0' encoding='utf-8' standalone='yes' ?><nlp>
  <version>1.1</version>
  <rawtext>开灯</rawtext>
  <confidence>96</confidence>
  <engine>local</engine>
  <result>
    <focus>cmd</focus>
    <confidence>99</confidence>
    <object>
      <cmd id="1">开灯</cmd>
    </object>
  </result>
</nlp>

313 bytes has been sent to socket.
Not started or already stopped.
waiting for data...
client disconnect
root@ubuntu: /mnt/hgfs/share/x86/bin#
```

```
root@ubuntu: /mnt/hgfs/share/tcp/1# ./tcp_xml_test
96000 bytes has been sent
313 bytes has been recv
313 bytes has been write to resul
confidence: 99
cmd: 开灯
id: 1
id=1
open light
root@ubuntu: /mnt/hgfs/share/tcp/1#
```

2. 语音引擎工作过程

```
1 root@ubuntu: /mnt/hgfs/share/x86/bin# ./asr_record_demo
2 Thu Dec 15 00:00:00 CST 2016
3 构建离线识别语法网络...
4 构建语法成功! 语法ID:cmd
5 离线识别语法网络构建完成, 开始识别...
6 wait for connecting ... // (1) 等待客户端连接
7 new connection: 127.0.0.1: 39786 // (2) 新的客户端[127.0.0.1: 39798]连接上
8 waiting for data... // (3) 等待客户端发送语音数据
9 [138]recv finished. get 96000 bytes // (4) 成功接收到客户端发送的语音数据, 大小为96KB
10 96000 bytes has been wirtten into cmd.pcm // (5) 将语音数据写入到本地cmd.pcm文件
11 Start Listening... // (6) 基于cmd.pcm文件分析语音数据
12 Result: // (7) 显示结果, 并xml格式进行展示
13 <?xml version='1.0' encoding='utf-8' standalone='yes' ?><nlp>
14 <version>1.1</version>
15 <rawtext>开灯</rawtext>
16 <confidence>98</confidence>
17 <engine>local</engine>
18 <result>
19 <focus>cmd</focus>
20 <confidence>99</confidence> // (8) 置信率(准确率) 99
21 <object>
22 <cmd id="1">开灯</cmd> // (9) 识别到命令词: 开灯, id值为1。
23 </object>
24 </result>
```

```

25 </nlp>
26
27 314 bytes has been sent to socket. // (10) 将识别的结果通过网络传输给客户端
28 Not started or already stopped.
29 waiting for data... // (11) 等待客户端新一轮语音数据
30 client disconnect // (12) 客户端断开连接

```

注:

科大讯飞语音引擎有个BUG, 如果ubuntu访问了外网, 在第二次识别的时候出现以下问题。

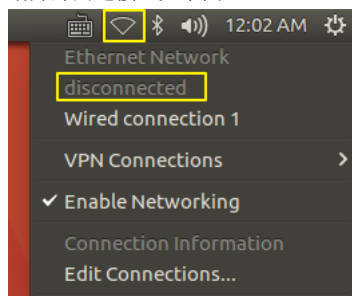
```

1 QISRGetResult failed! error code: 11203
2
3 Recognizer error 11203
4
5 sr_stop_listening failed! error code:11203

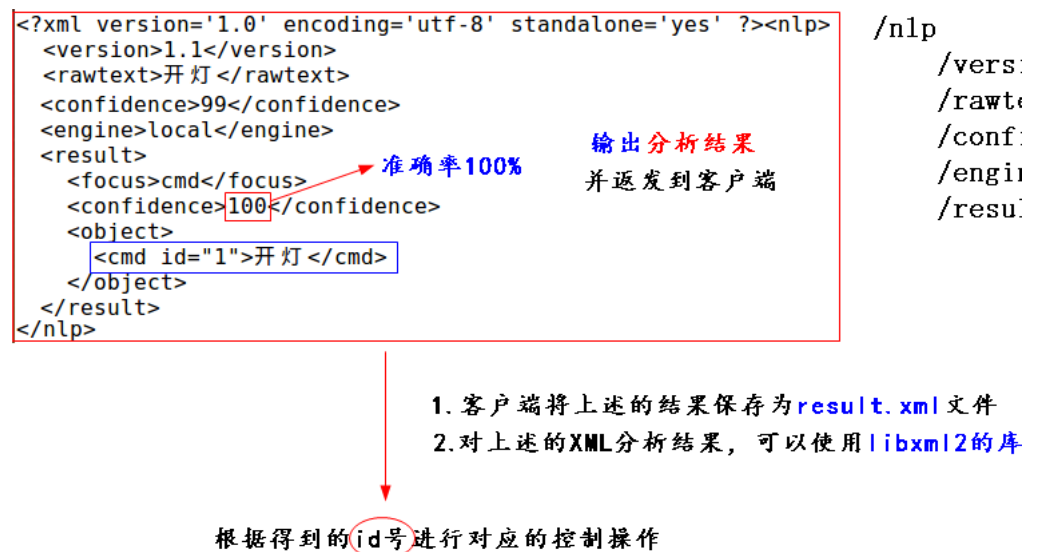
```

解决方法:

断开外网连接, 如下图。



3. 梳理思路



id号来源于语音引擎中的cmd.bnf文件, 详细路径在x86/bin目录下

```

#BNF+IAT 1.0 UTF-8;

!grammar cmd;
!slot <cmd>;
!start <cmdstart>;

<cmdstart>:<cmd>;

<cmd>: 开灯!id(1) | 关灯!id(2) | 切换下一张图片!id(3) | 切换上一张图片!id(4) | 小欣同学!id(5)

```

提示: 如果修改了cmd.bnf文件中的内容, 需要重新编译语音引擎!

