**一、ALSA**

首先下载群文件里面的如下内容：

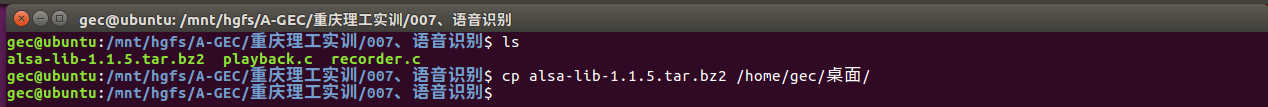


移植ALSA：

高级Linux声音架构的简称，它在Linux操作系统上面提供了音频和MIDI的支持

**1、解压alsa**

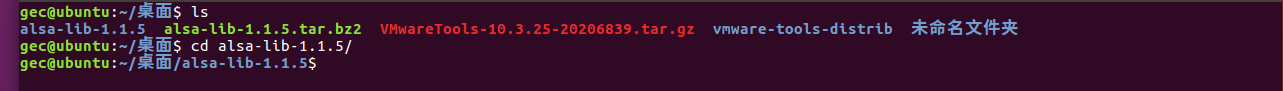
**1.1 将 alsa-lib-1.1.5.tar.bz2 拷贝到Ubuntu的桌面，cp alsa-lib-1.1.5.tar.bz2 /home/gec/桌面/**



**1.2 切换到桌面，解压压缩包，tar -xvf alsa-lib-1.1.5.tar.bz2**

image.png

**1.3 进入文件夹alsa-lib-1.1.5，cd alsa-lib-1.1.5**



**2、编译配置**

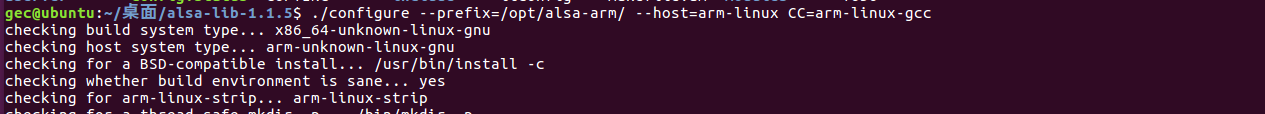
**（1）ARM版本**

执行 ./configure --prefix=/opt/alsa-arm --host=arm-linux CC=arm-linux-gcc --disable-python

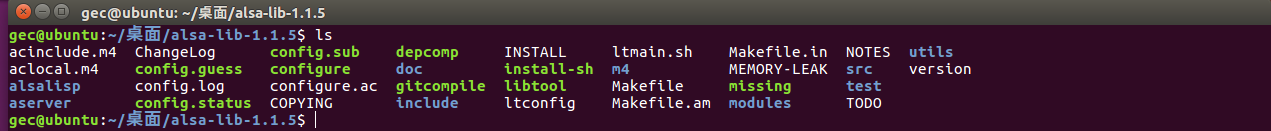
--prefix=/opt/alsa-arm/：指定安装目录

--host=arm-linux：指定库的运行平台，比如交叉编译是ARM平台

CC：指定编译



编译之后查看文件ls

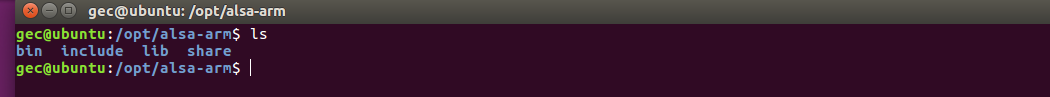


主要需要的是Makefile文件，执行makefile进行编译

image.png

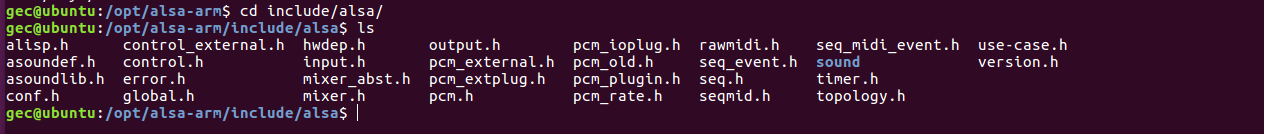
执行命令 sudo make install 进行安装

安装完成之后，切换到 /opt/alsa-arm/ 目录下面

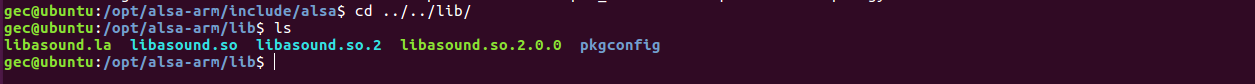


查看是否编译好

①查看include/alsa文件夹，有没有头文件



②查看lib目录下面，有没有库文件



**（2）Ubuntu版本**

由于上面已经配置了ARM版本，配置Ubuntu之前先删除之前的配置

在/alsa-lib-1.1.5目录下

执行 make clean

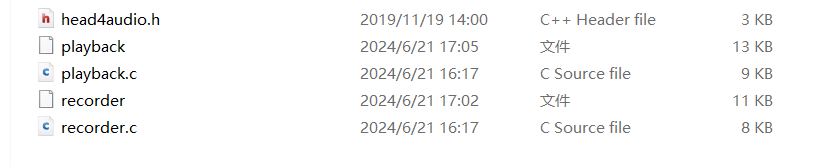
执行 ./configure --prefix=/opt/alsa-ubuntu/ CC=gcc

进行 make 编译

执行 sudo make install 进行安装

切换到 /opt/alsa-ubuntu 目录下查看是否编译好，也是查看头文件include跟库文件lib目录下有没有文件

**3、源码设计**



head4audio.h：头文件

playback.c：播放音频源码

recorder.c：录制音频

录音过程：

打开录音文件.wav

打开PCM，做抽样量化编码

配置录音文件格式和参数

开始录音

回到共享目录，编译以上两个文件：

arm-linux-gcc recorder.c -o recorder -I /opt/alsa-arm/include/ -L /opt/alsa-arm/lib/ -lasound

arm-linux-gcc playback.c -o playback -I /opt/alsa-arm/include/ -L /opt/alsa-arm/lib/ -lasound

-I /opt/alsa-arm/include/：指定头文件所在路径

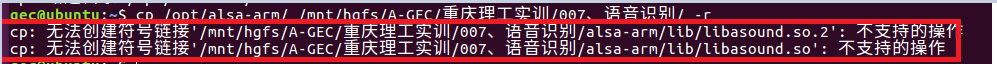
-L /opt/alsa-arm/lib/：指定库文件所在路径

-lasound：要连接的库文件的名字

**4、移植ARM开发板**

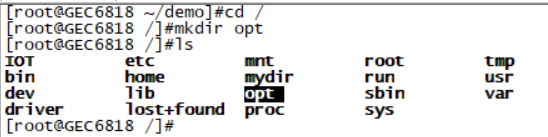
4.1把ARM版本的alsa库移植到开发板

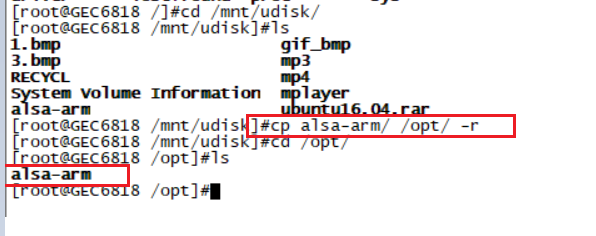
将Ubuntu里面的 /opt/alsa-arm 这个目录下面的所有文件全部拷贝到开发板 /opt/alsa-arm



忽略上述错误，因为这两个是软连接，无法拷贝

在拷贝之前，先查看开发板是否具有 /opt 目录（一般没有），需要手动创建该文件夹





4.2 恢复刚才失效的两个软连接

ln -s /opt/alsa-arm/lib/libasound.so.2.0.0 /opt/alsa-arm/l

ib/libasound.so.2

ln -s /opt/alsa-arm/lib/libasound.so.2.0.0 /opt/alsa-arm/l

ib/libasound.so

4.3 设置环境变量

将以下语句写入到启动文件 /etc/profile当中

image.png

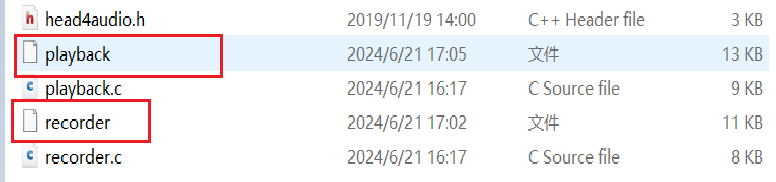
按下i进入输入模式，输入以下语句

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/opt/alsa-arm/lib/

按下 esc 键，进入底行模式，输入 :wq 保存退出

输入reboot重启开发板

上传可执行文件playback、recorder



./recorder test.wav 3：录制音频3表示录制事件

./playback test.wav：播放音频

image.png

**二、语音识别测试**

**2.1 SDK与API**

**2.1.1 什么是API**

假设有两个开发人员A与开发人员B，人员A开发软件A，人员B开发软件B，有一天开发人员B对A说我想要用一下你某个功能但是我不想去看源代码，A就对B说我将实现这段功能代码进行封装成一个函数，你照着我说的流程去做，就可以将代码添加到自己的项目当中，此时在这封装的函数就称为API

**2.1.2 什么是SDK**

为了提高开发效率，减少程序员的工作量就诞生了SDK（软件开发工具包），辅助开发某一类软件的文档、使用示例和工具等等一些集合，就称为SDK

**2.2 SDK与API有什么区别**

SDK相当于一个虚拟的程序包，在这个程序包当中有一份做好的软件功能，但是程序包处于一个几乎全封闭的状态，只有一个小小的接口可以联通外界，那这个接口就是API

举例：买了一杯奶茶，需要一根吸管来喝里面奶，这个奶茶就相当于SDK，吸管就相当于API

**2.3 SDK下载**

**2.3.1进入官网下载语音包源码**

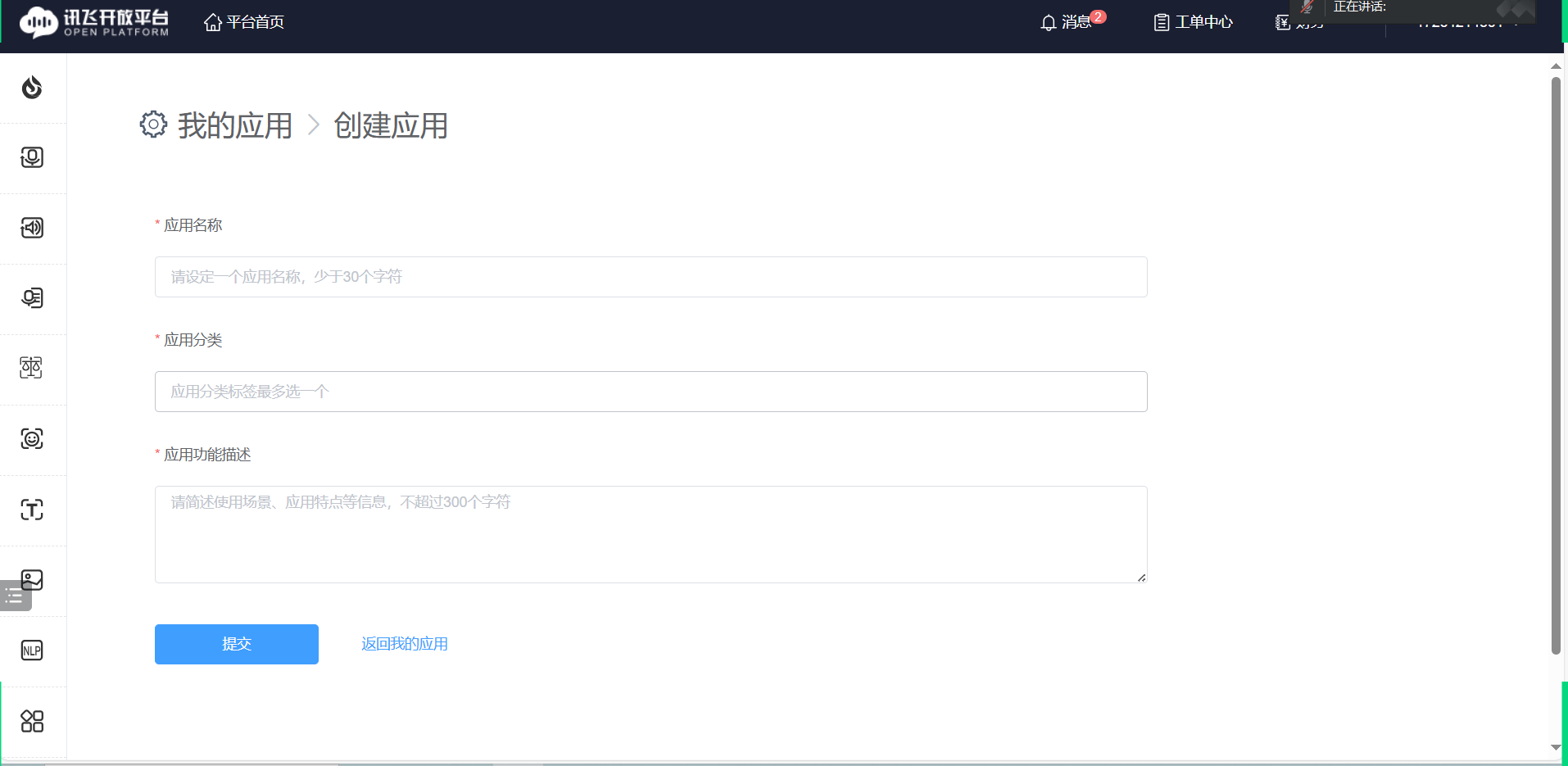
[www.xfyun.cn](http://www.xfyun.cn)

**2.3.2 注册登录**



**2.3.3 创建应用**

控制台里面，创建应用



**2.3.4 下载SDK**





下载后，会得到如下压缩包

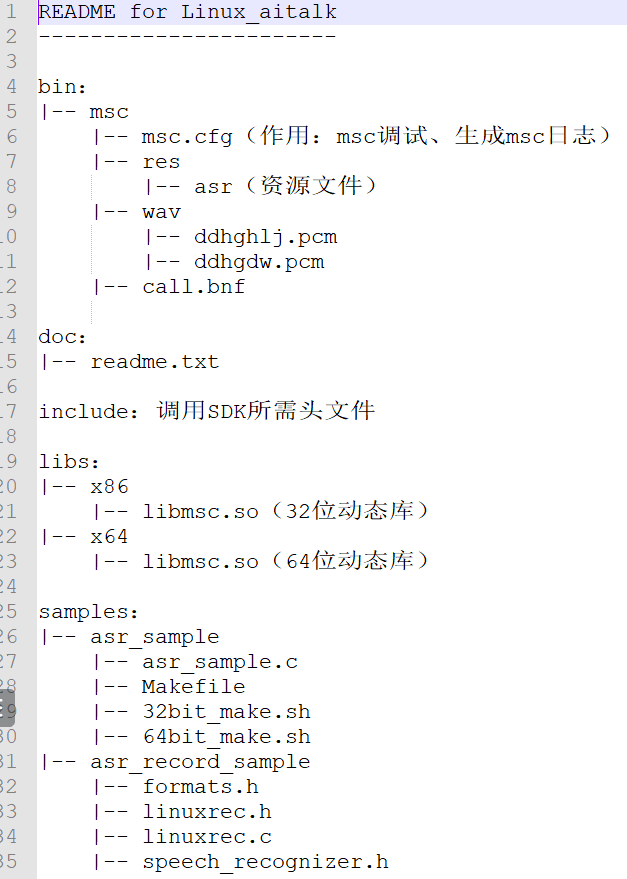
image.png

**2.3.5 使用测试代码**

解压压缩包：



阅读文档：README，了解每个文件夹的功能



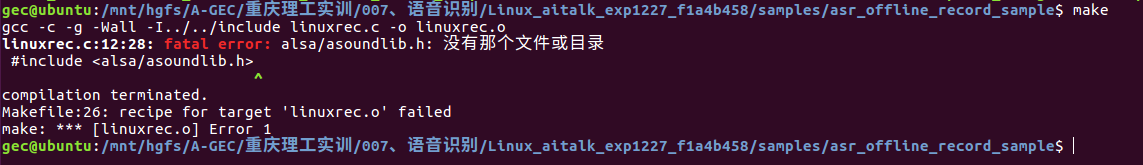
**2.3.6 编译测试代码**

1、进入示例程序路径

image.png

2、执行make

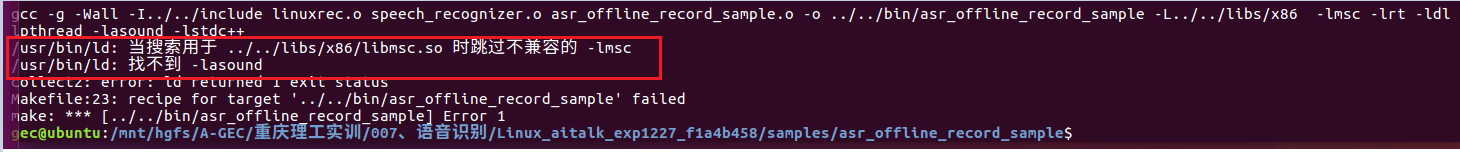
问题1：提示工程里面没有alsa头文集



解决办法：将我们编译好的ALSA相关头文件拷贝到当前工程目录

sudo cp /opt/alsa-ubuntu/include/\* ../../include/ -r

问题2：找不到-lasound库



解决办法：自己指定编译的库

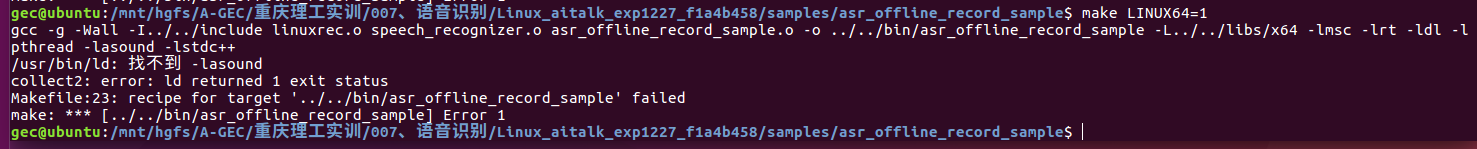
make LINUX64=1

或者

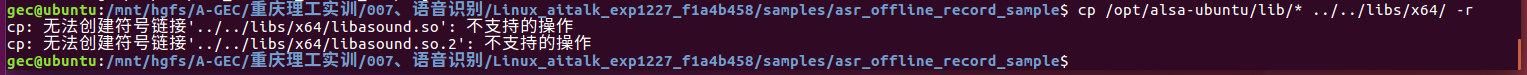
./64bit\_make.sh

区别：./64bit\_make.sh会清空之前编译的文件，重新编译，make LINUX64=1不会清空之前编译的东西

问题3：



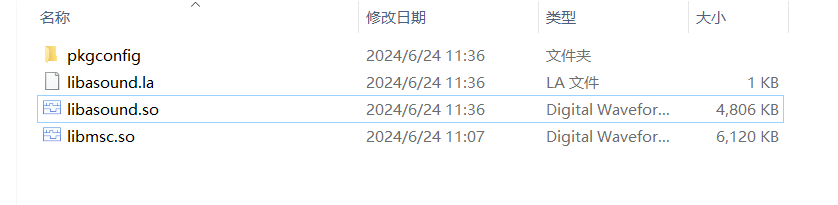
解决办法：将我们编译好的ALSA库文件拷贝到当前工程的库文件夹下面去



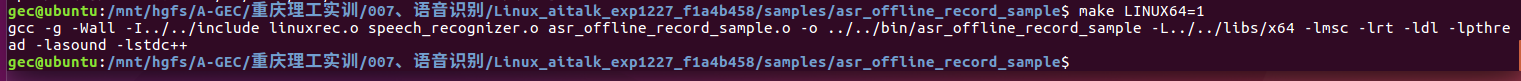
不支持的操作，无法创建符合链接，不管它

更改库文件名：

将 lib/x64 里面的libasound.so.2.0.0更改成libasound.so



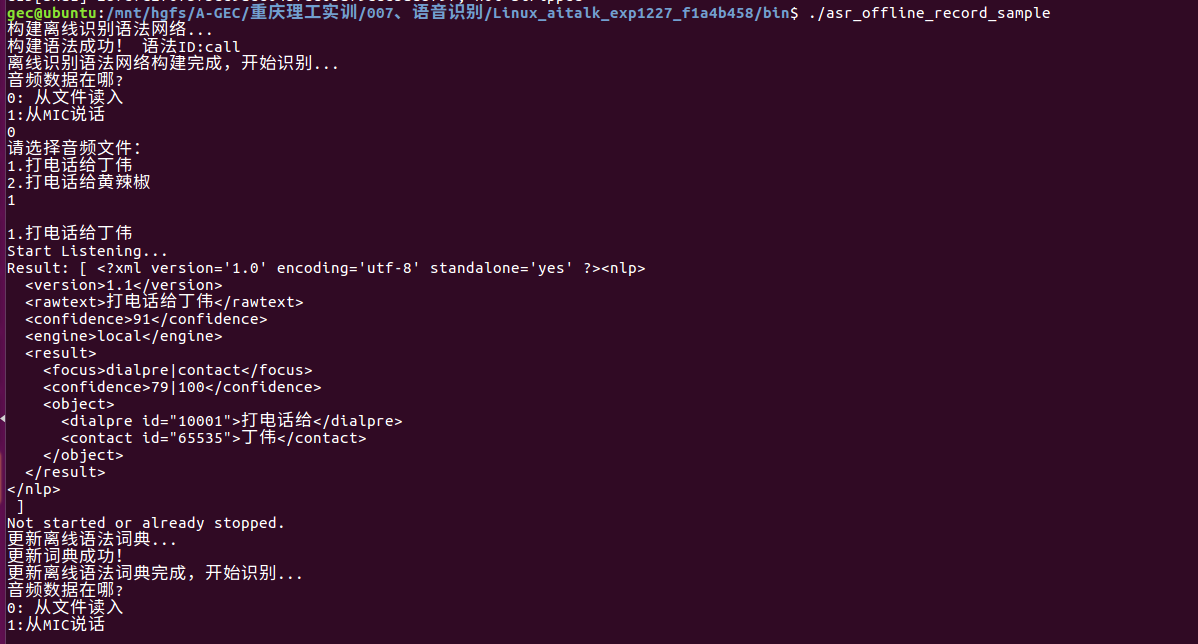
再去编译就没有报错：



**2.3.7 运行测试代码**

进入示例工程的bin目录下面：cd ../../bin

运行程序：./asr\_offline\_record\_sample



识别结果如下（xml格式）：

Result: [ <?xml version='1.0' encoding='utf-8' standalone='yes' ?><nlp>

<version>1.1</version>

<rawtext>打电话给丁伟</rawtext>

<confidence>91</confidence>

<engine>local</engine>

<result>

<focus>dialpre|contact</focus>

<confidence>79|100</confidence>

<object>

<dialpre id="10001">打电话给</dialpre>

<contact id="65535">丁伟</contact>

</object>

</result>

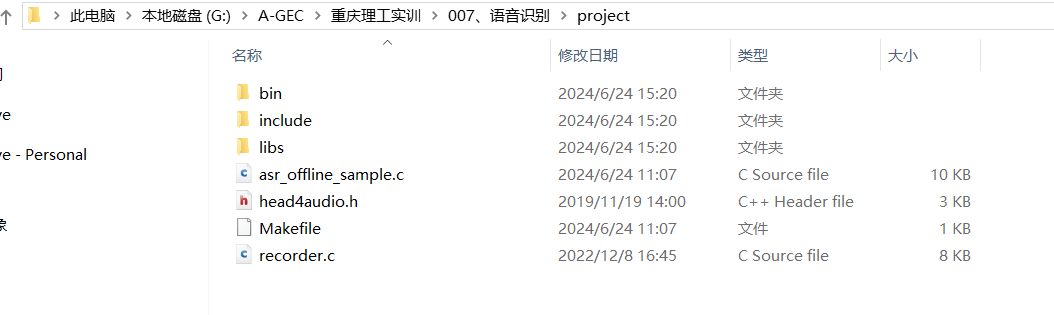
</nlp>

]

**三、语音识别移植ARM板**

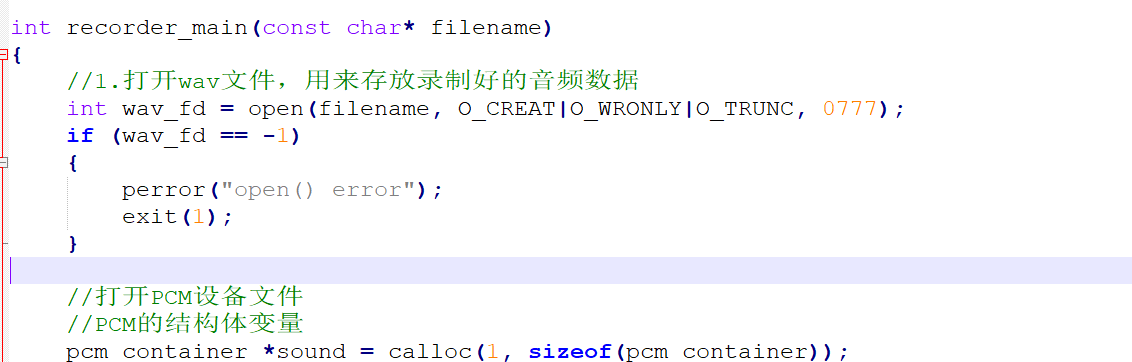
**3.1 创建工程模板**

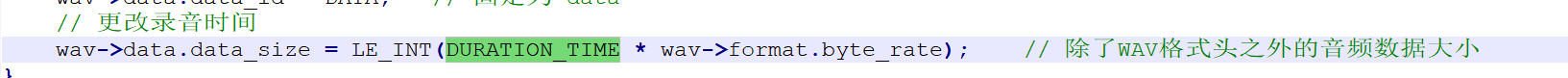
在自己创建工程目录当中，拷贝以下文件



**3.2 修改recorder.c**

修改recorder.c的主函数，将它改成一个普通函数，修改文件路径和录音时间





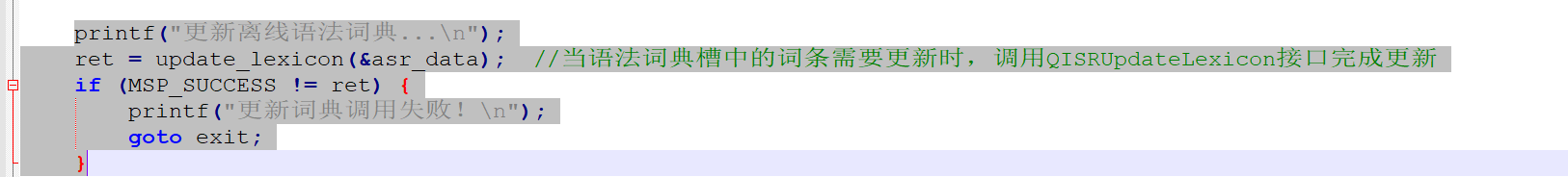
**3.3 修改asr\_offline\_sample.c代码**

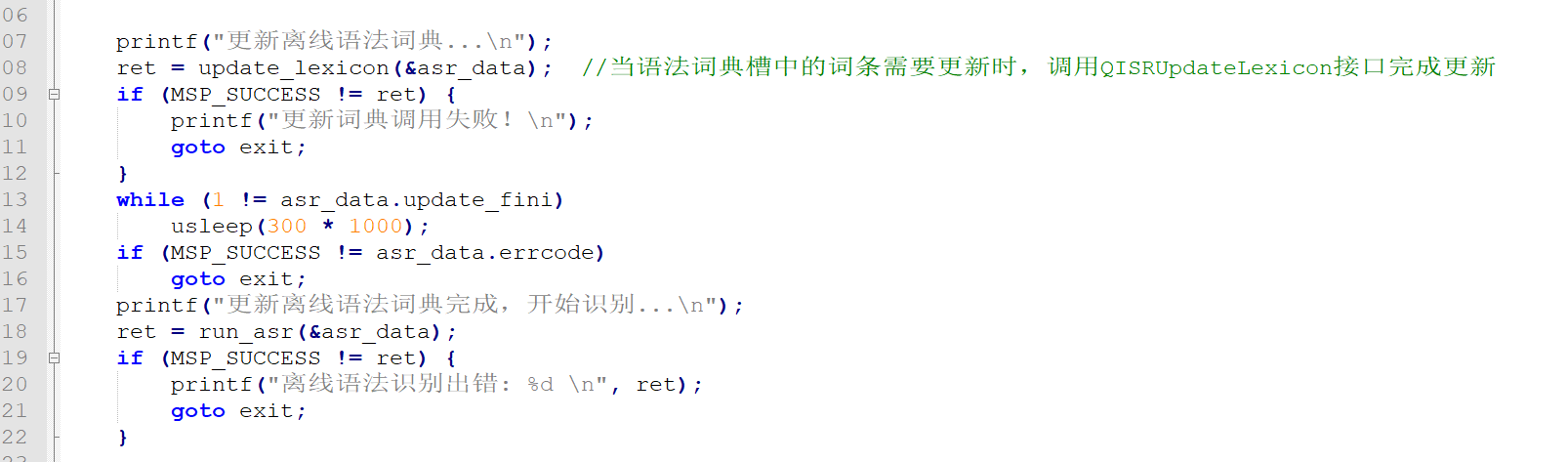
**3.3.1 修改main函数，成为一个普通函数**

1、修改主函数名字

image.png

2、删除以下代码：





3、给函数run\_asr添加参数

image.png

image.png

**3.3.2 删掉不需要的一些函数**

1、删掉const char \*get\_audio\_file(void);函数

2、删掉int update\_lexicon(UserData \*udata);函数

3、删掉int update\_lex\_cb(int ecode, const char \*info, void \*udata)函数

**3.3.3 保留不变的函数**

1、 int build\_grm\_cb(int ecode, const char \*info, void \*udata)

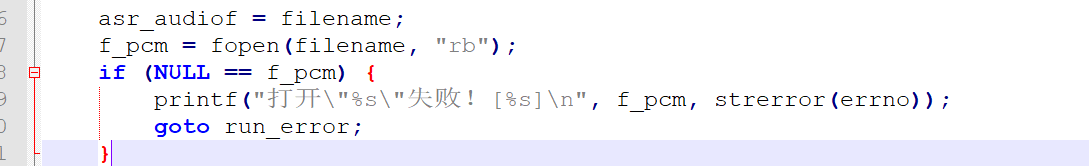
2、int build\_grammar(UserData \*udata)

**3.3.4 修改函数run\_asr（UserData \*udata）**

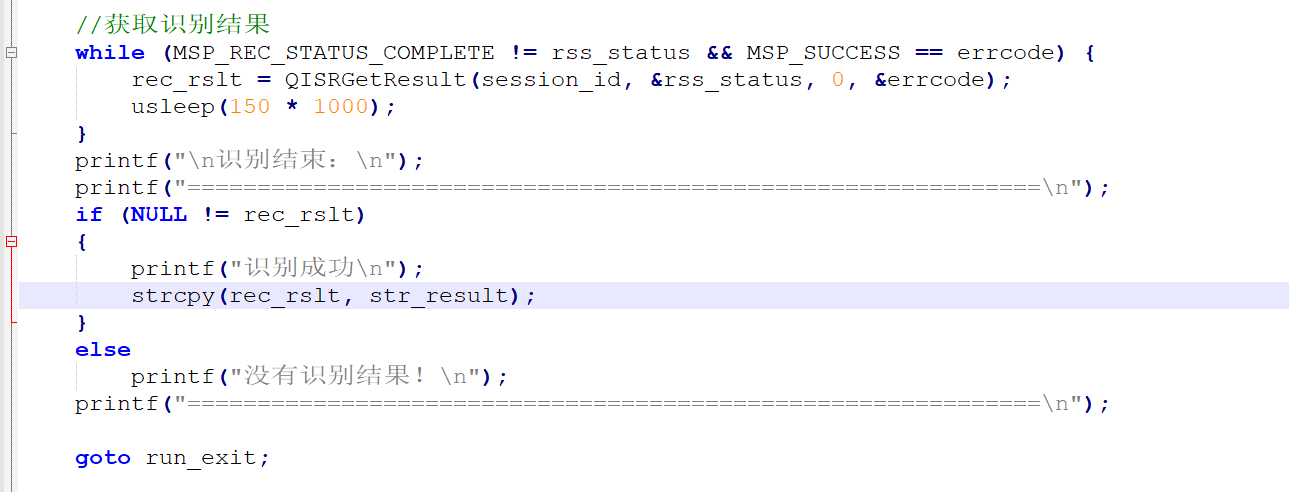
修改的第一个地方

image.png

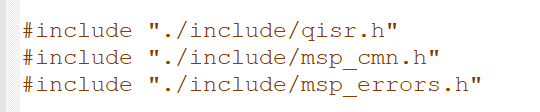
修改的第二个地方



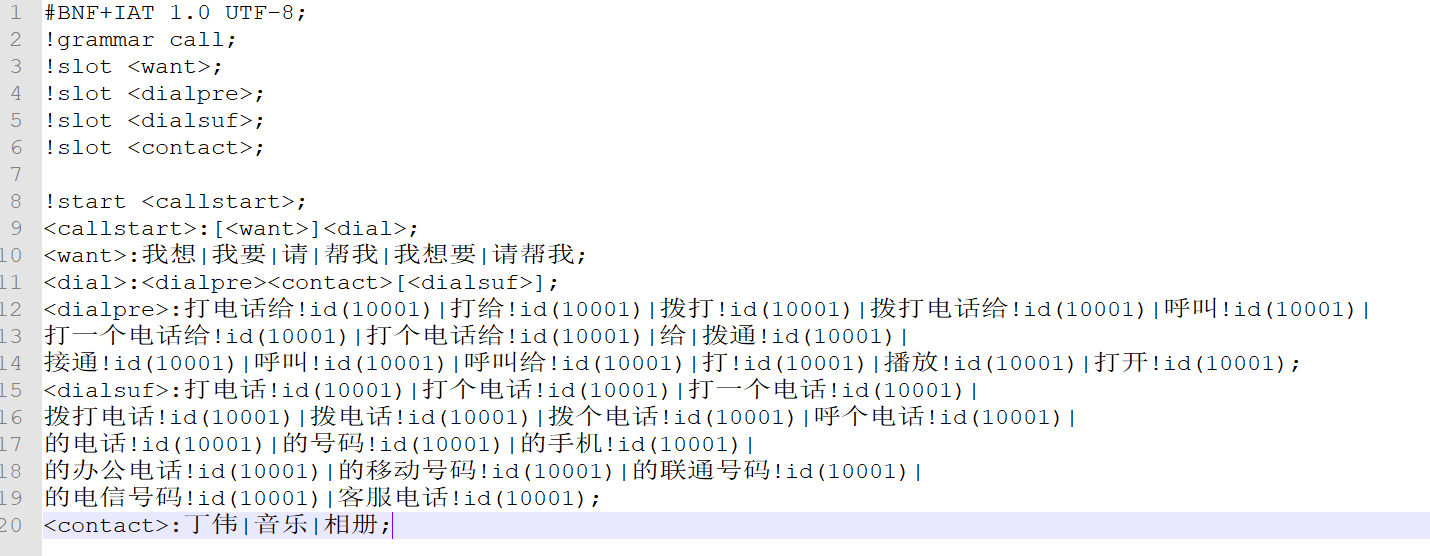
修改的第三个地方：



修改的第三个地方：



**3.4 添加call.bnf词条**



**3.5 自定义main函数**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int recorder\_main(char \*filename);

int asr\_main(const char \*filename, char \*str\_result);

int main(int argc, char const \*argv[])

{

//录音

char \*filename = "test.wav";

recorder\_main(filename);

//定义一个字符串存放识别结果

char str\_result[1024] = {"NULL"};

asr\_main(filename, str\_result);

if(strcmp(str\_result, "NULL")!=0)

{

printf("识别结果\n");

printf("%s\n", str\_result);

if (strstr(str\_result, "音乐") != NULL)

{

system("madplay faded.mp3 &");

}

if (strstr(str\_result, "相册") != NULL)

{

//播放之前录的声音

printf("我在\n");

}

if (strstr(str\_result, "维斯") != NULL)

{

//播放之前录的声音

printf("hello sir\n");

}

}

return 0;

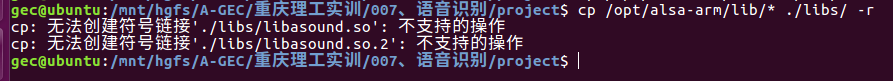
}

**3.5 编译工程**

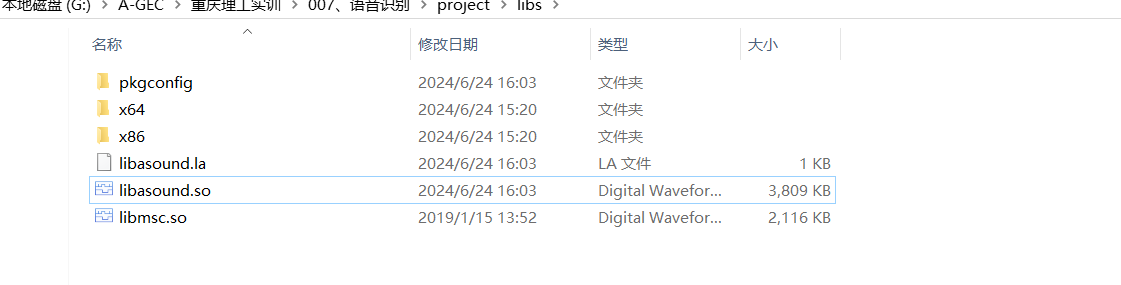
1、将ALSA头文件拷贝到工程目录当中

image.png

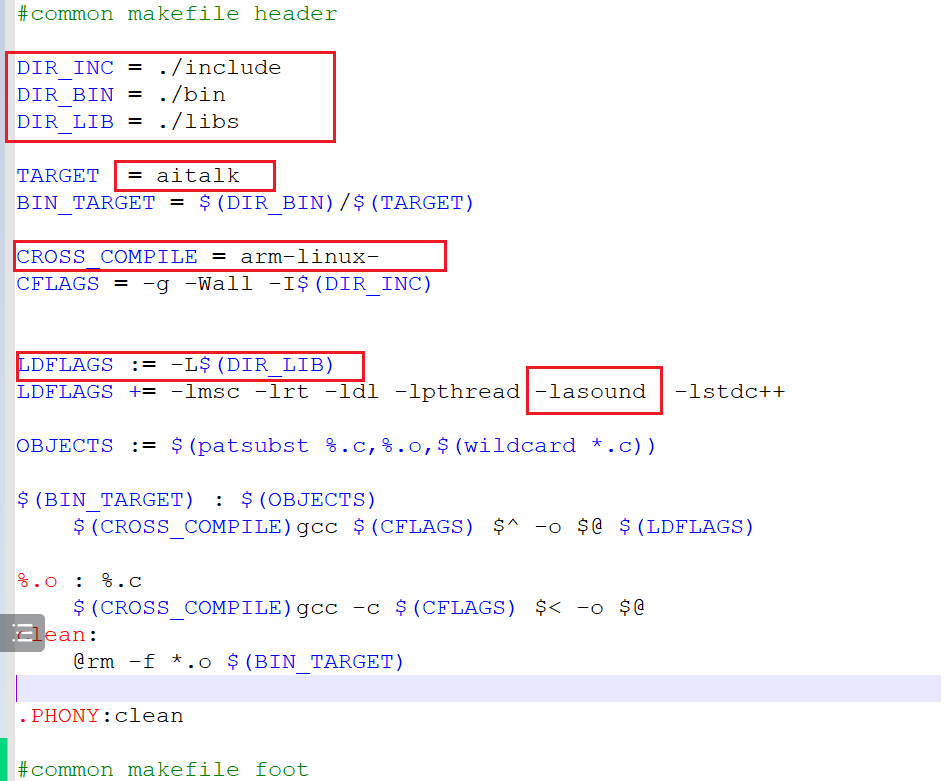
2、拷贝ALSA的库文件



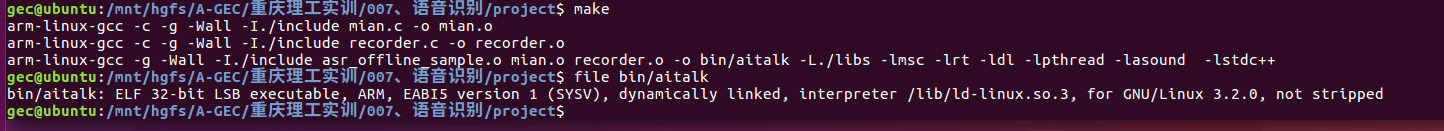
3、libmsc.so这个库要自己放进来，更改libasound.so.2.0.0为libasound.so



4、修改Makefile



5、编译工程

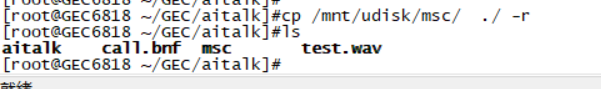


**3.6 下载开发板**

1、下载可执行文件在开发版里面

2、将词条call.bnf下载到开发板，注意一定要与可执行文件在同一路径

3、将msc这个文件夹下载到开发板，注意一定要与可执行文件在同一路径



4、将库libmsc.so拷贝开发板的 /bin 目录下

5、运行程序

