介于网上资料多有缺陷，而科大讯飞所提供的事例也有漏洞，所以没法强行移植

**一、环境配置**

**1. 安装 libasound**

**1.1.安装指令**

apt-get install libasound2-dev

**1.2. 验证安装**

ls /usr/lib/x86\_64-linux-gnu/libasound.so

**2.安装sox**

sox 是一个非常强大的音频处理工具，它支持录音、播放、格式转换等功能

**2.1 安装指令**

sudo apt-get install sox

**2.2 验证安装**

sox --version

**3.安装libsox-fmt-all**

**3.1 安装指令**

sudo apt-get install libsox-fmt-all

**3.2 验证安装**

无，这是sox的扩展库，无验证

**4.科大讯飞SKD文件下载**

**一、科大讯飞平台下载开放包及语音验证**

提示：本文针对的是语音识别中的语音识别（流式版）

讯飞平台：[科大讯飞平台链接](https://console.xfyun.cn/app/myapp)

需要先注册账号，然后创建自己的应用才能够对开放包进行下载（每个开放包都有自己的APPID号，这个很重要！）

创建应用的界面

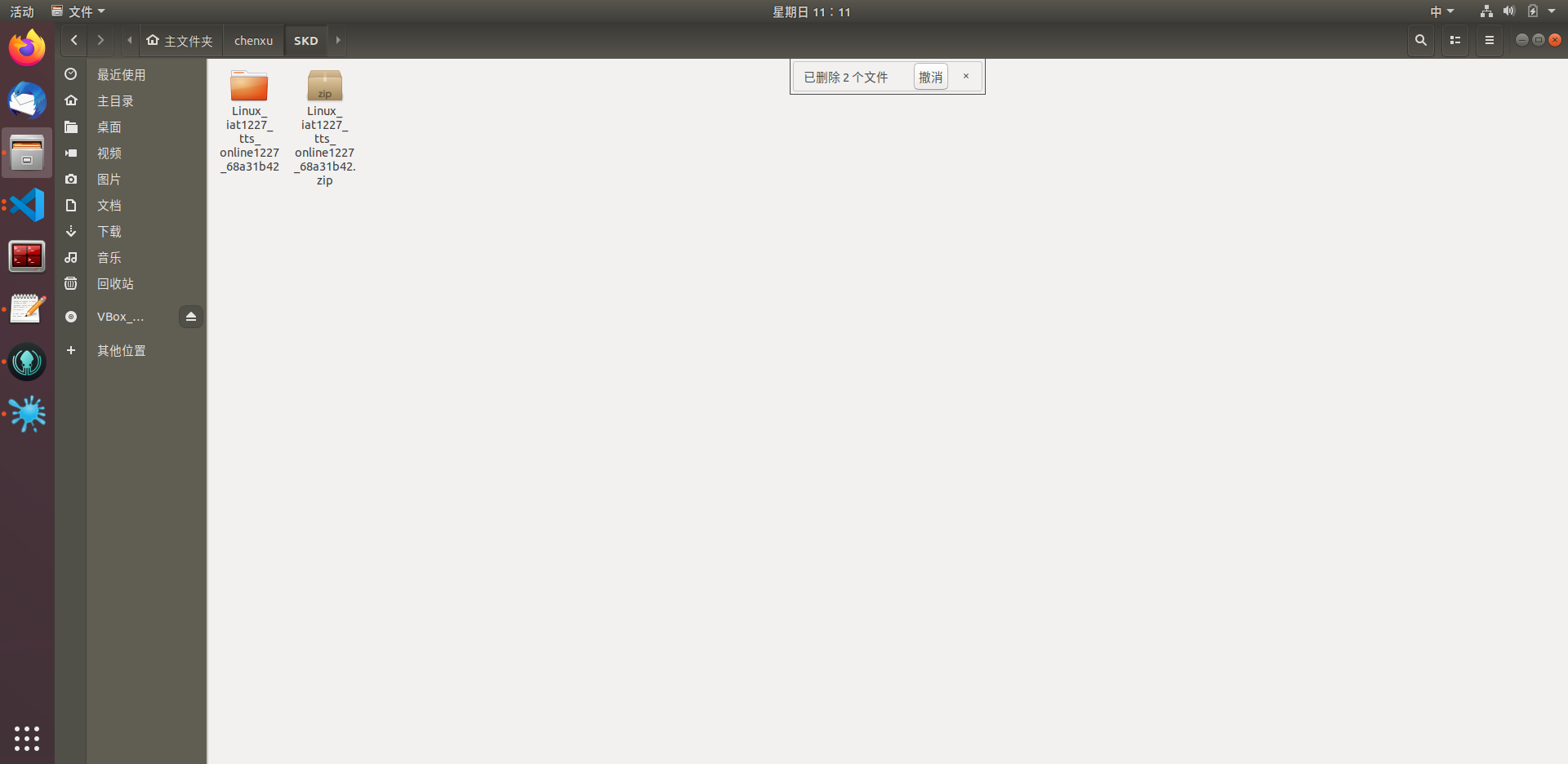


下载的包（随便点一个，因为点了选择的时候，也是默认安卓）





将开放包下载到linux系统中，然后解压

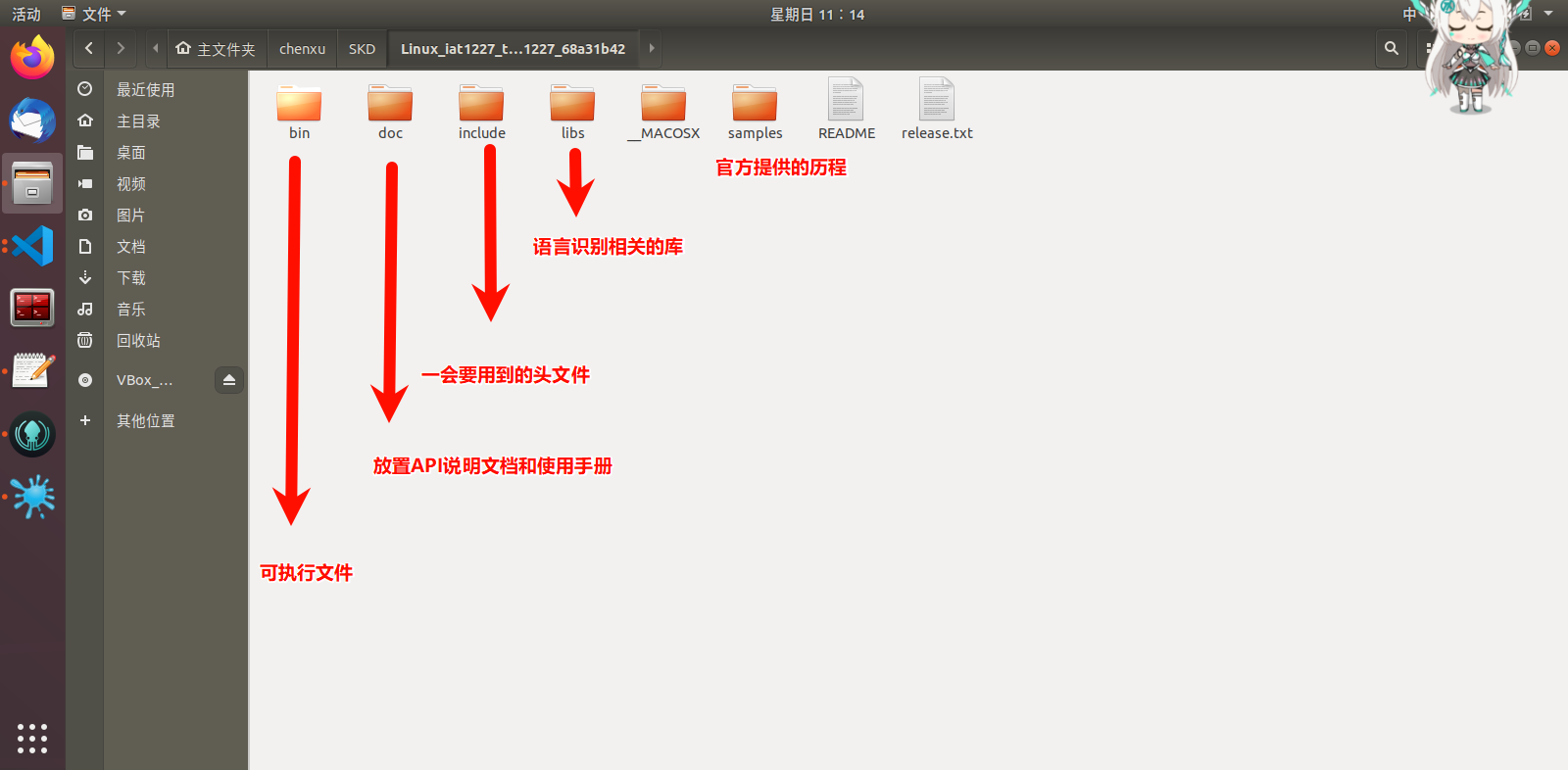


ubuntu一般是不（默认）支持zip格式解压的，所以请你下载一个解压工具，这里推荐 unzip

sudo apt update

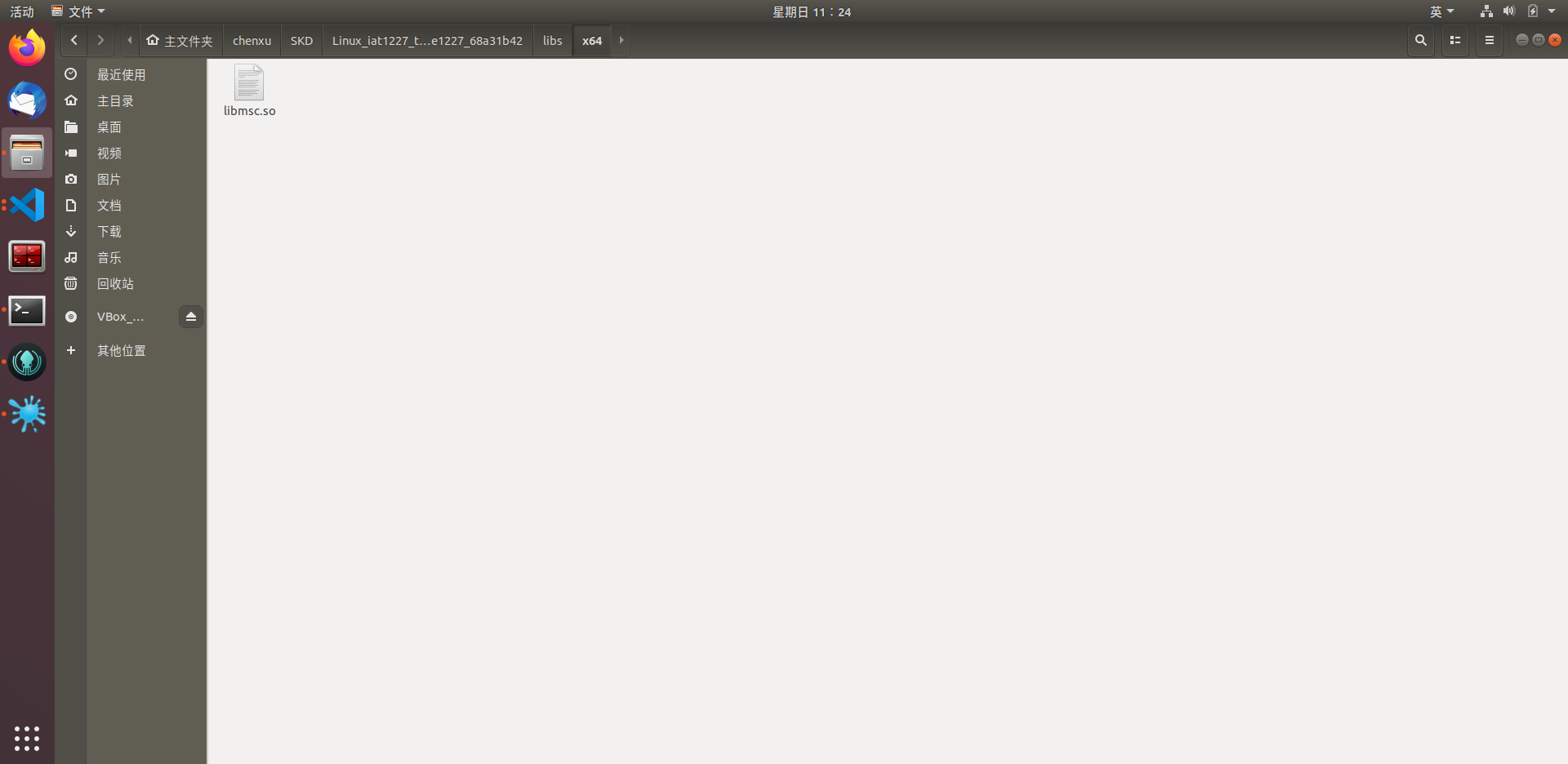
sudo apt install unzip

这是包中各个文件及其里面的内容



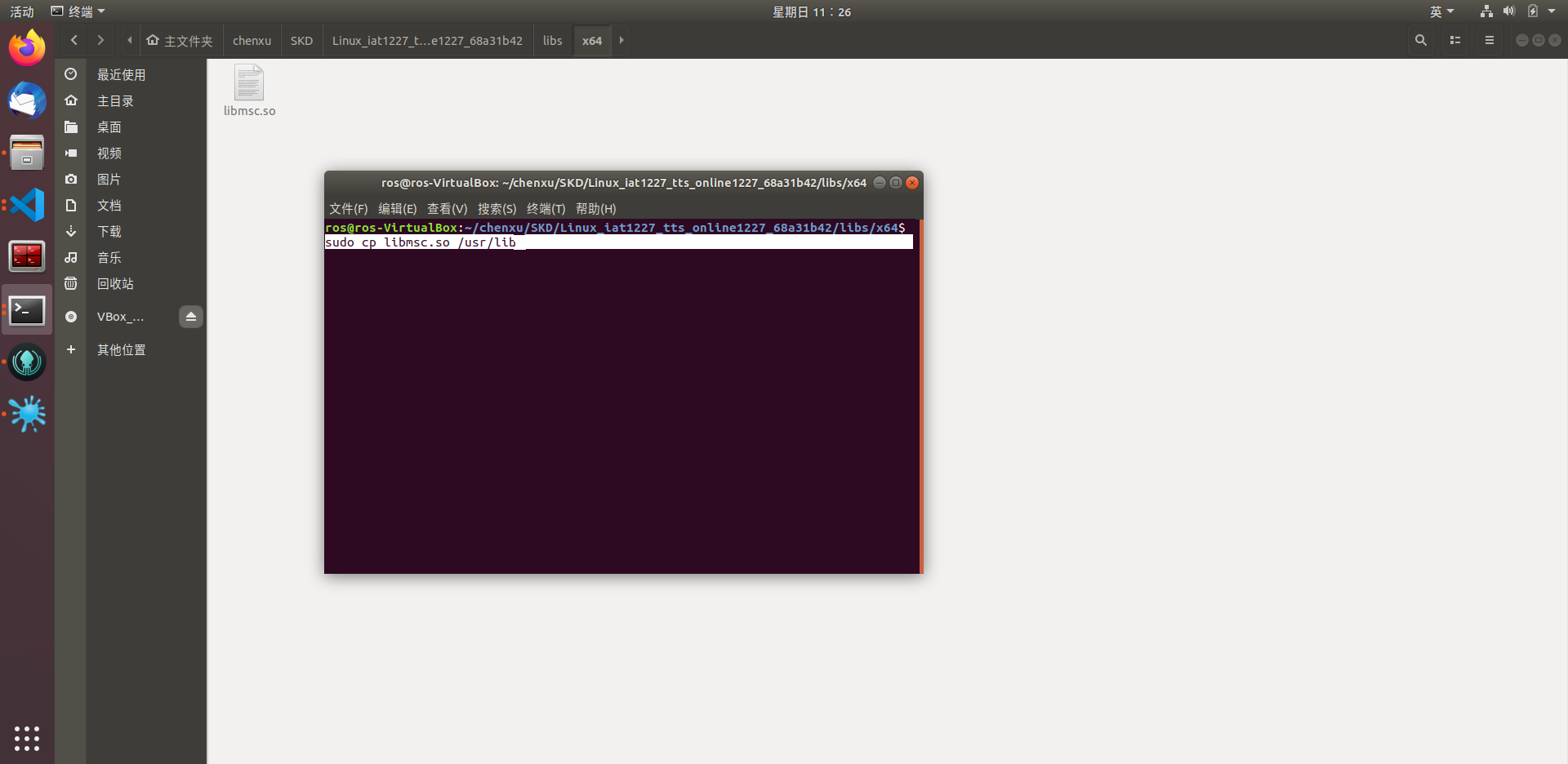
重点要用到的是samples和libs里面的这些包

首先我们先进入libs/x64文件下

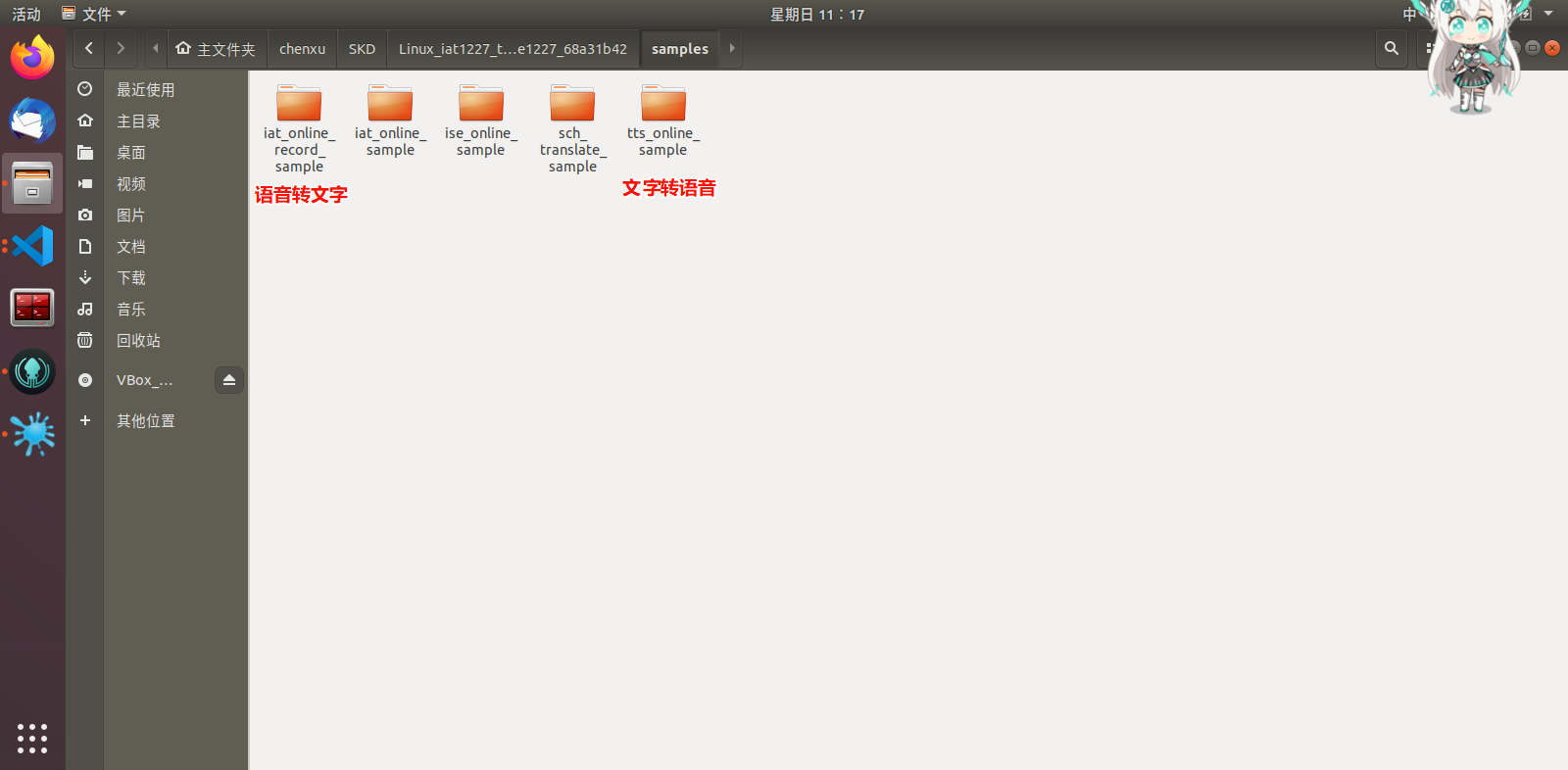


接着将其中的.so文件复制到/usr/lib文件当中

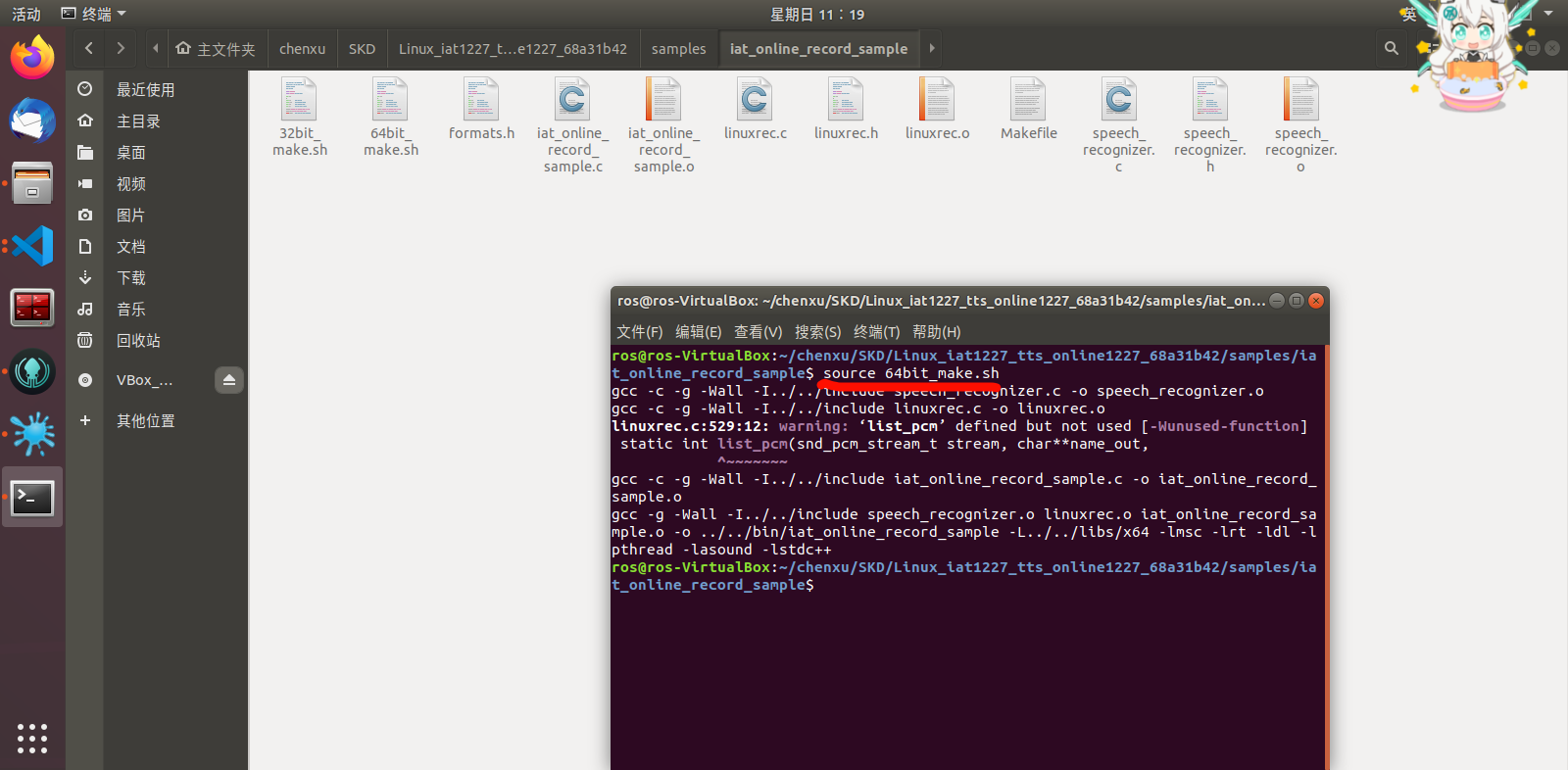
指令：sudo cp libmsc.so /usr/lib

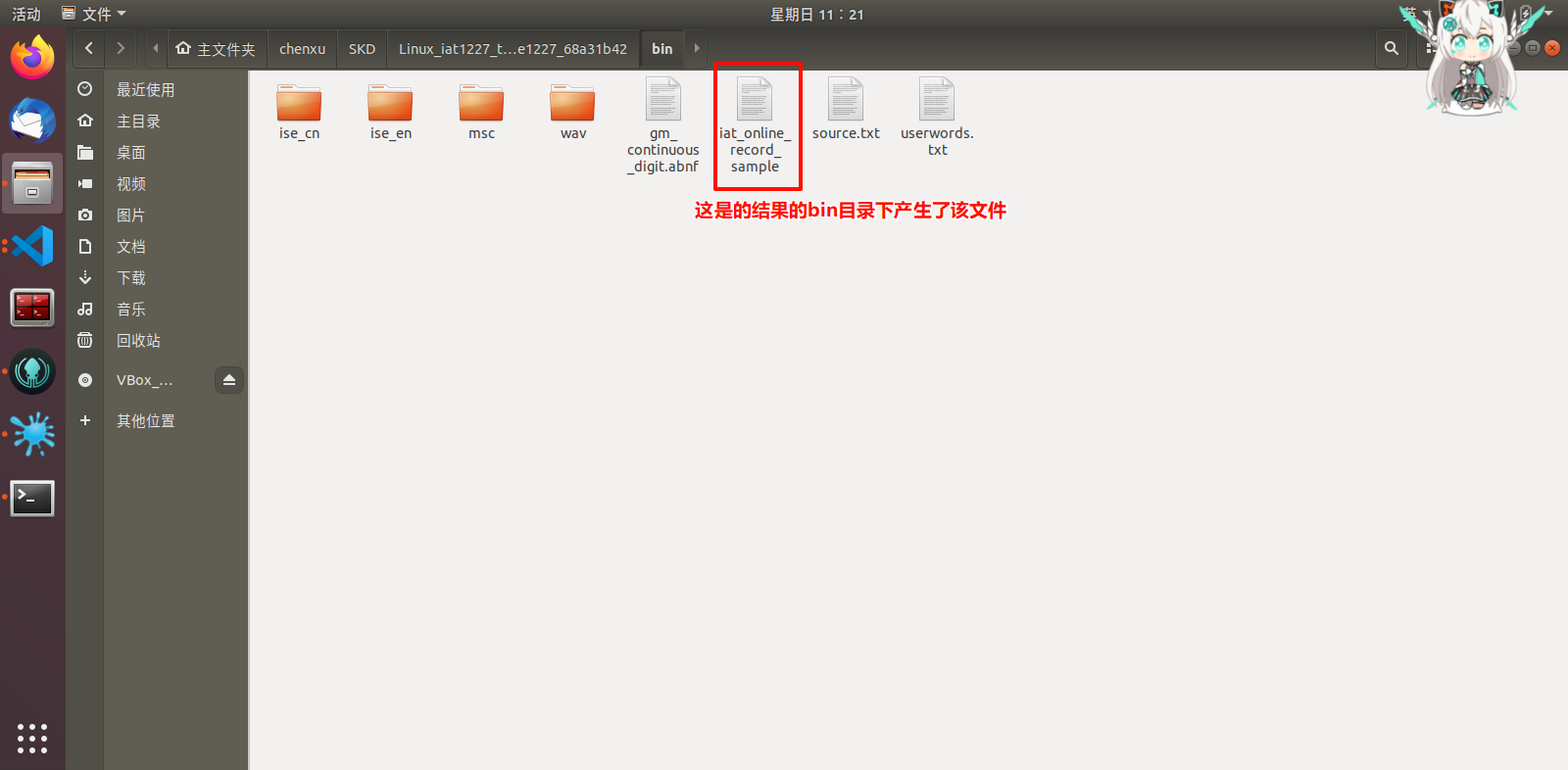


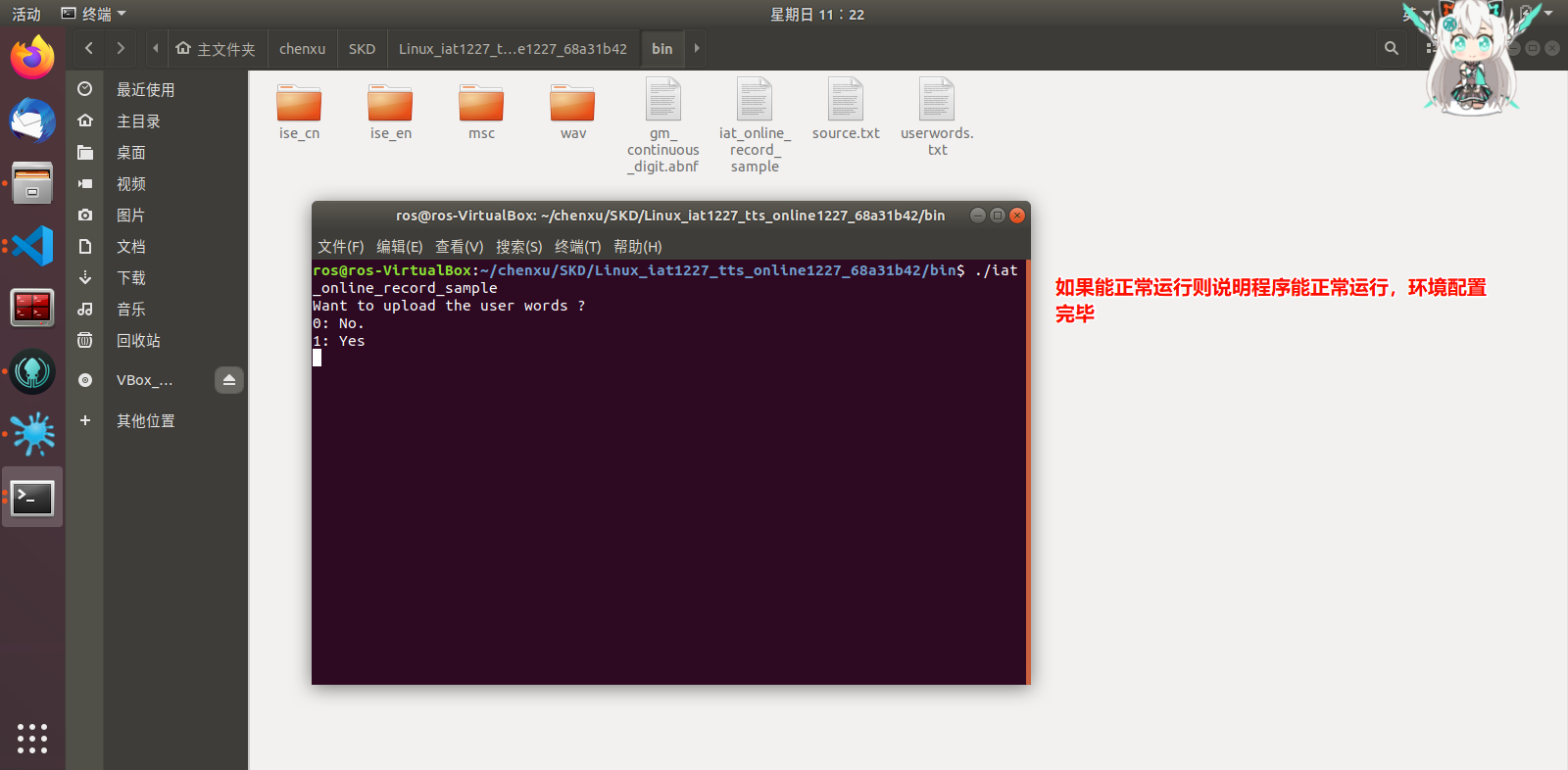
接着我们再打开samples文件夹当中



进入iat\_online\_record\_sample功能包下，打开终端，执行.sh脚本文件进行编译







**二、ROS环境下的语音识别与输出**

**1.ROS语音识别**

**1.创建工作空间并初始化**

mkdir -p 自定义空间名称/src

cd 自定义空间名称

catkin\_make

上述命令，首先会创建一个工作空间以及一个 src 子目录，然后再进入工作空间调用 catkin\_make命令编译。

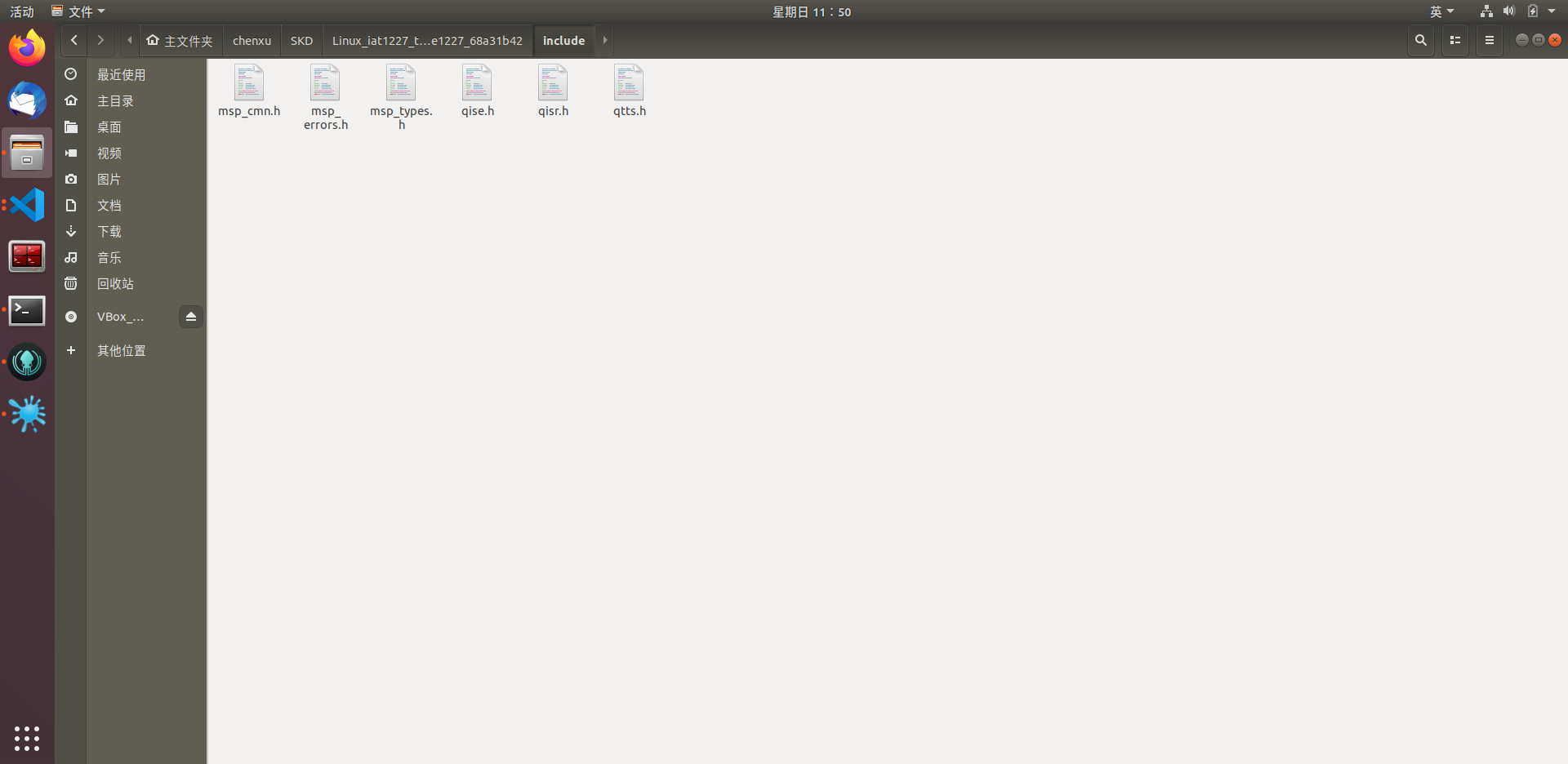
**2.进入 src 创建 ros 包并添加依赖**

cd src

catkin\_create\_pkg 自定义ROS包名 roscpp rospy std\_msgs

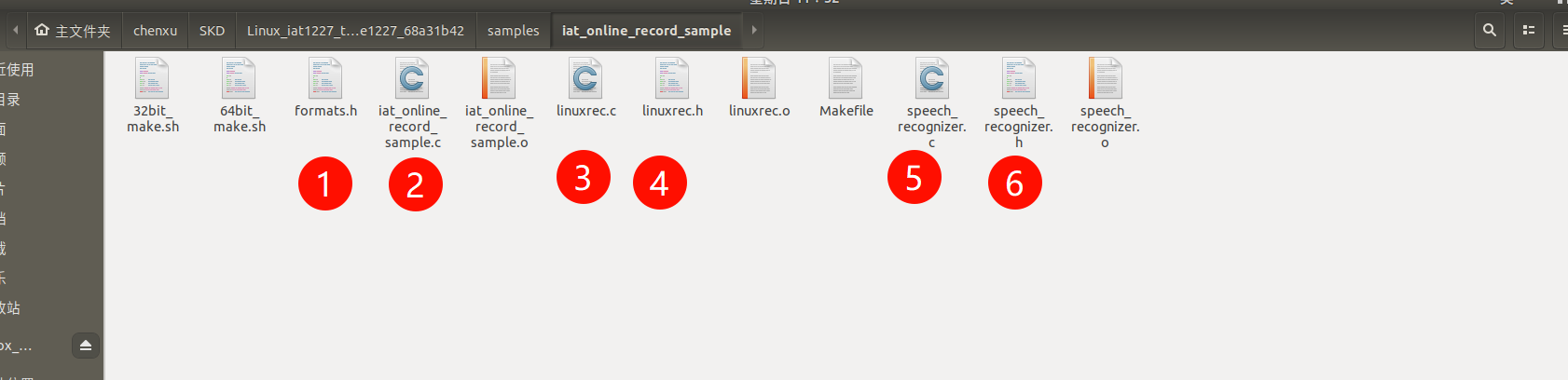
**3.头文件的引用**

**3.1首先将SDK的include文件下的所有文件，复制到你的功能包下的include文件夹，注意最好只是在include文件夹里，而不是include/功能包名的文件夹下（因为这样能省下一些步骤）**



**3.2接着将这六个文件复制到include文件夹下（.c文件放到src文件夹下）**

注意再复制完后，最好把这三个.c文件修改为.cpp文件，不用害怕会报错，因为cpp文件是c文件的超集，是可以兼容的，我们做这一步是为了方便ros系统的编译



**4.xf\_asr文件**

**原先的文件iat\_online\_record\_sample太长，不方便记忆，所以我将这个文件改名于xf\_asr，并为了兼容其ros系统，并作出了以下调整。**

**/\***

**\*** 语音听写(iFly Auto Transform)技术能够实时地将语音转换成对应的文字。

**\*/**

**#i**nclude <stdlib.h>

**#i**nclude <stdio.h>

**#i**nclude <string.h>

**#i**nclude <unistd.h>

**#i**nclude "qisr.h"

**#i**nclude "msp\_cmn.h"

**#i**nclude "msp\_errors.h"

**#i**nclude "speech\_recognizer.h"

**#i**nclude <iconv.h>

**#i**nclude "ros/ros.h"

**#i**nclude "std\_msgs/String.h"

**#d**efine FRAME\_LEN 640

**#d**efine BUFFER\_SIZE 4096

**in**t wakeupFlag = 0;

**in**t resultFlag = 0;

**st**atic char \*g\_result = NULL;

**st**atic unsigned int g\_buffersize = BUFFER\_SIZE;

**st**atic void show\_result(char \*string, char is\_over)

**{**

resultFlag = 1;

// 只输出最终的识别结果

printf("%s\n", string);

**}**

**vo**id on\_result(const char \*result, char is\_last)

**{**

if (result) {

size\_t left = g\_buffersize - 1 - strlen(g\_result);

size\_t size = strlen(result);

if (left < size) {

g\_result = (char\*)realloc(g\_result, g\_buffersize + BUFFER\_SIZE);

if (g\_result)

g\_buffersize += BUFFER\_SIZE;

else {

printf("mem alloc failed\n");

return;

}

}

strncat(g\_result, result, size);

show\_result(g\_result, is\_last); // 识别完成后直接显示结果

}

**}**

**vo**id on\_speech\_begin()

**{**

if (g\_result) {

free(g\_result);

}

g\_result = (char\*)malloc(BUFFER\_SIZE);

g\_buffersize = BUFFER\_SIZE;

memset(g\_result, 0, g\_buffersize);

**}**

**vo**id on\_speech\_end(int reason)

**{**

if (reason == END\_REASON\_VAD\_DETECT)

printf("\nSpeaking done\n");

else

printf("\nRecognizer error %d\n", reason);

**}**

**/\*** demo recognize the audio from microphone \*/

**st**atic void demo\_mic(const char\* session\_begin\_params)

**{**

int errcode;

int i = 0;

struct speech\_rec iat;

struct speech\_rec\_notifier recnotifier = {

on\_result,

on\_speech\_begin,

on\_speech\_end

};

errcode = sr\_init(&iat, session\_begin\_params, SR\_MIC, &recnotifier);

if (errcode) {

printf("speech recognizer init failed\n");

return;

}

errcode = sr\_start\_listening(&iat);

if (errcode) {

printf("start listen failed %d\n", errcode);

}

/\* demo 10 seconds recording \*/

while (i++ < 2)

sleep(1);

errcode = sr\_stop\_listening(&iat);

if (errcode) {

printf("stop listening failed %d\n", errcode);

}

sr\_uninit(&iat);

**}**

**vo**id WakeUp(const std\_msgs::String::ConstPtr& msg)

**{**

printf("waking up\r\n");

usleep(700\*1000);

wakeupFlag = 1;

**}**

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

**{**

// 初始化ROS

ros::init(argc, argv, "voiceRecognition");

ros::NodeHandle n;

ros::Rate loop\_rate(10);

// 声明Publisher和Subscriber

ros::Subscriber wakeUpSub = n.subscribe("voiceWakeup", 1000, WakeUp);

ros::Publisher voiceWordsPub = n.advertise<std\_msgs::String>("voiceWords", 1000);

ROS\_INFO("Waiting for wake-up signal...");

while (ros::ok())

{

// 语音识别唤醒

if (wakeupFlag){

ROS\_INFO("Wakeup...");

int ret = MSP\_SUCCESS;

const char\* login\_params = "appid = , work\_dir = .";

const char\* session\_begin\_params =

"sub = iat, domain = iat, language = zh\_cn, "

"accent = mandarin, sample\_rate = 16000, "

"result\_type = plain, result\_encoding = utf8";

ret = MSPLogin(NULL, NULL, login\_params);

if(MSP\_SUCCESS != ret){

MSPLogout();

printf("MSPLogin failed , Error code %d.\n",ret);

}

printf("Demo recognizing the speech from microphone\n");

demo\_mic(session\_begin\_params);

wakeupFlag = 1;

MSPLogout();

}

// 语音识别完成，发布结果

if(resultFlag)

{

resultFlag = 0;

std\_msgs::String msg;

msg.data = g\_result;

voiceWordsPub.publish(msg); // 发布识别结果

}

ros::spinOnce();

loop\_rate.sleep();

}

MSPLogout(); // Logout...

return 0;

**}**

**5.头文件的应用（vscode）**

这儿我们用的是vscode作为编译器

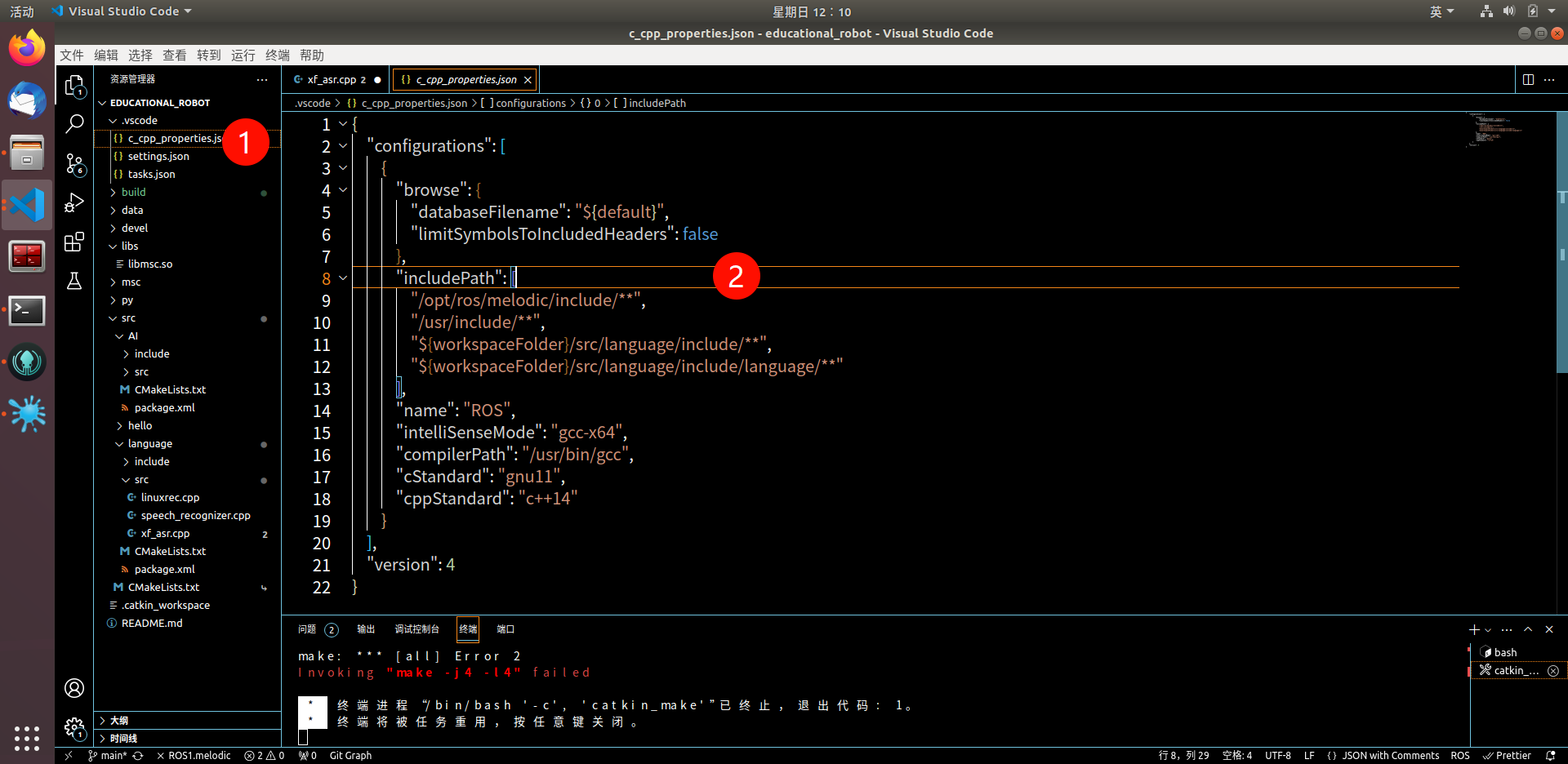
现在的你必然是无法通过编译的，而且程序中依然有大量的红色波浪线，这是因为程序找不到你头文件的所在，所以我们来到.vscode的文件夹目录下，点开c\_cpp\_properties.json

接着就在includePath在加入这么一段

"${workspaceFolder}/src/ <你的功能包名> /include/\*\*",

"${workspaceFolder}/src/<你的功能包名>/include/<你的功能包名>/\*\*"

如图所示（注意逗号位置，否则任然会报错）



**6.CMakeLists.txt的配置**

现在你编译是不是依然还在报错，ros的环境配置就是有点麻烦，现在我们打开你的功能包下的CMakeLists.txt文本，并依次修改文件。主要修改部分为：

find\_package(ALSA REQUIRED)

catkin\_package(

CATKIN\_DEPENDS roscpp rospy std\_msgs

)

include\_directories(

include

${catkin\_INCLUDE\_DIRS}

${ALSA\_INCLUDE\_DIRS}

)

#这一步很重要，否则你的文件就不能编译通过了，因为他找不到函数

add\_executable(xf\_asr src/xf\_asr.cpp

src/speech\_recognizer.cpp

src/linuxrec.cpp)

#这一步也是，如果没做就无法识别语音

target\_link\_libraries(xf\_asr

${catkin\_LIBRARIES}

/usr/lib/libmsc.so

${ALSA\_LIBRARIES}

)

**7.bug修复**

怎么说呢，你做完上一步，基本也就大差不差了，但问题是科大讯飞的代码有些问题，尽管测试时能顺利通过，但在vscode里，你编译的时候，还是会爆出大量的错误。这就源于科大讯飞所提供的代码中的语法错误了。

这一点我记得不太清，挺多的，只能说几个要命的地方，其余小bug，建议配合AI和你的经验解决（最好还是靠着你的经验，AI有点蠢，分不清一级和二级指针，揪着一个错误答案，死都不改，主打就是虚心认错，死不悔改）。

**7.1Linuxrec文件**

**7.1.1函数list\_pcm，建议修改成以下这样**

原代码有则const char 变 char的权限扩大，也有开辟空间不做强制类型转换，将void类型传给char的神奇操作。

static int list\_pcm(snd\_pcm\_stream\_t stream, char\*\* name\_out, char\*\* desc\_out)

{

void \*\*hints, \*\*n;

char \*\*name, \*\*descr;

char \*io;

const char \*filter;

int cnt = 0;

int i = 0;

if (snd\_device\_name\_hint(-1, "pcm", &hints) < 0)

return 0;

n = hints;

cnt = get\_pcm\_device\_cnt(stream);

if (!cnt) {

goto fail;

}

name\_out = (char\*\*)malloc((1 + cnt) \* sizeof(char\*));

desc\_out = (char\*\*)malloc((1 + cnt) \* sizeof(char\*));

if (\*name\_out == NULL)

goto fail;

if (\*desc\_out == NULL)

goto fail;

(\*name\_out)[cnt] = NULL;

(\*desc\_out)[cnt] = NULL;

name = name\_out;

descr = desc\_out;

filter = (stream == SND\_PCM\_STREAM\_CAPTURE) ? "Input" : "Output"; // const char\*

while (\*n != NULL && i < cnt) {

\*name = snd\_device\_name\_get\_hint(\*n, "NAME");

\*descr = snd\_device\_name\_get\_hint(\*n, "DESC");

io = snd\_device\_name\_get\_hint(\*n, "IOID");

if (\*name == NULL || (io != NULL && strcmp(io, filter) != 0)) {

if (\*name) free(\*name);

if (\*descr) free(\*descr);

} else {

if (\*descr == NULL) {

\*descr = (char\*)malloc(128); // 分配足够空间

memset(\*descr, 0, 128); // 初始化内存

}

name++;

descr++;

i++;

}

if (io != NULL)

free(io);

n++;

}

snd\_device\_name\_free\_hint(hints);

return cnt;

fail:

free\_name\_desc(name\_out);

free\_name\_desc(desc\_out);

snd\_device\_name\_free\_hint(hints);

return -1;

}

**7.2speech\_recognizer文件**

**7.2.1函数skip\_space，建议修改成以下这样**

static const char \* skip\_space(const char \*s)

{

while (s && \*s != ' ' && \*s != '\0')

s++;

return s;

}

**8.运行**

这时候基本就没什么问题了，运行即可，对了建议修改xf\_asr文件下语音输入时的if (wakeupFlag)为if（1）。之后运行ros节点便好了。

对了，这个代码，还有这一个缺陷，我还没改，就是说一句话就会生成一个文件，随便说一阵就两百个文件堆起来了，不用猜就能知道，用不了多久，他就能塞满整个空间，（主要是节点关闭了，文件也不会自动销毁，主打的就是个内存泄露拉满），有空在修复吧，毕竟这不是我写的代码，我只是做了个整合，还得再看看。