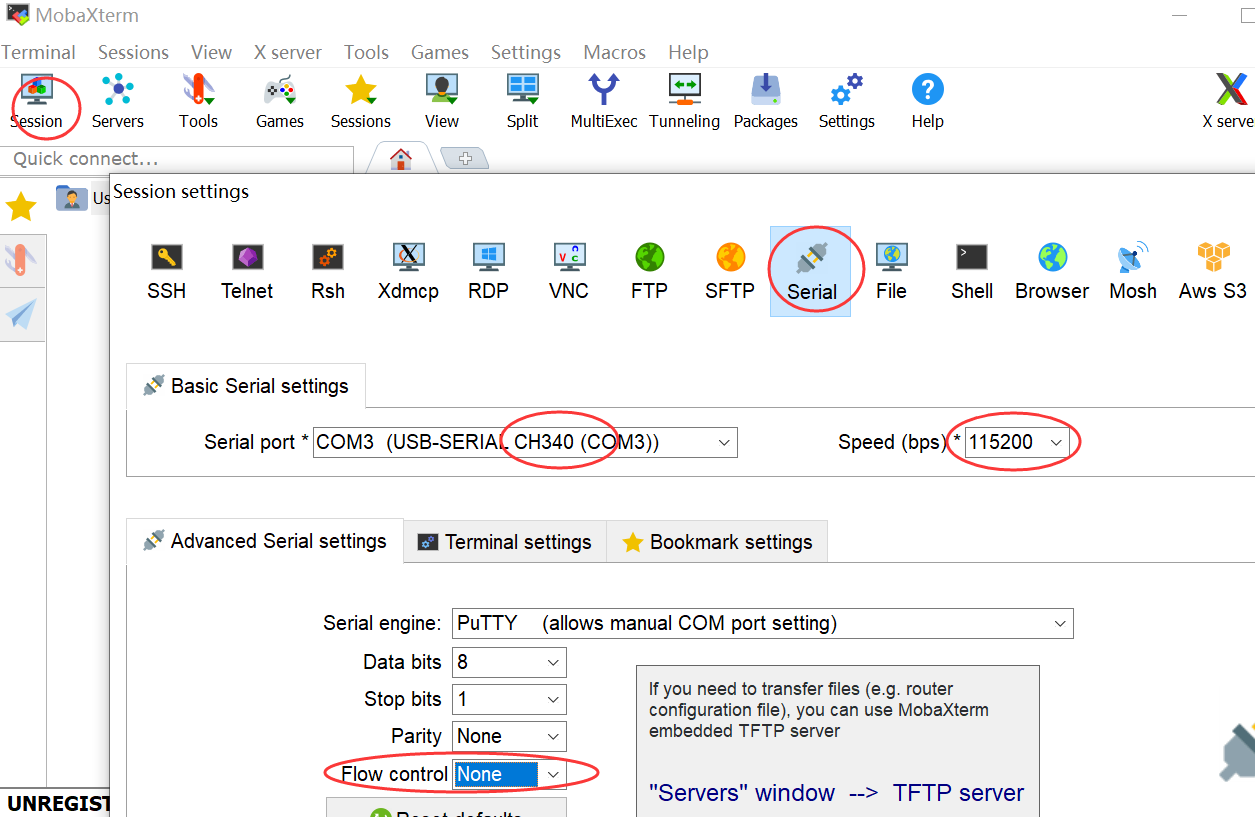
# 镜像烧写

1. debian11-fs4412-opencv-2.2.zip解压得到debian11-fs4412-opencv-2.2.img
2. 以管理员权限运行烧录软件：balenaEtcher-Portable-1.18.4.exe
3. 将SD卡插入读卡器，连接到开发电脑
4. 使用烧写工具将下载的镜像文件烧写到SD卡（TF卡）中
5. 将SD卡（TF卡）插入FS4412的卡槽中，并将开发板上的OM拨码开关设置为从SD卡启动。



# 启动A9单板

1. 安装CH341SER.EXE
2. 运行MobaXterm\_Personal\_23.1.exe
3. 电脑通过串口连接的A9开发板



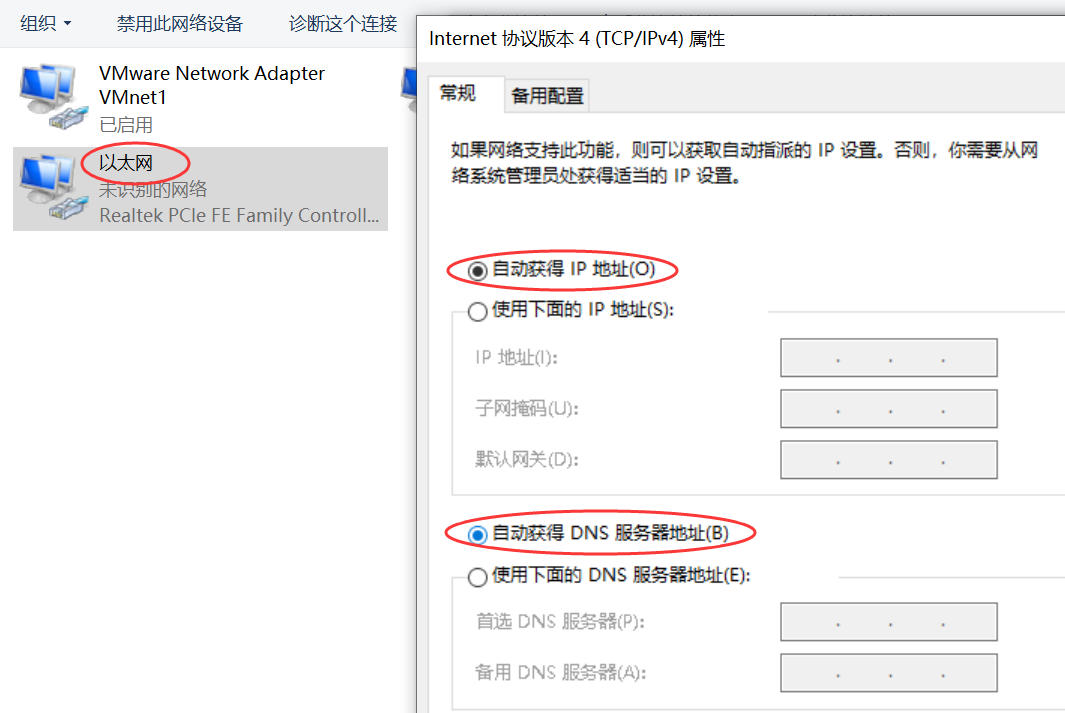
1. 操作系统启动后，可以在串口上使用root用户登录，登录密码为hqyj
2. 系统第一次启动后，建议执行/root目录中的setup.sh脚本，扩大根文件系统到整个SD卡（TF卡），并生成机器ID。否则，可能出现根文件系统空间不足和IP地址冲突等问题。



1. 安装完成后，开发板会自动关机，但不会自动关闭电源。出现reboot: System halted信息后，可以关闭开发板电源。

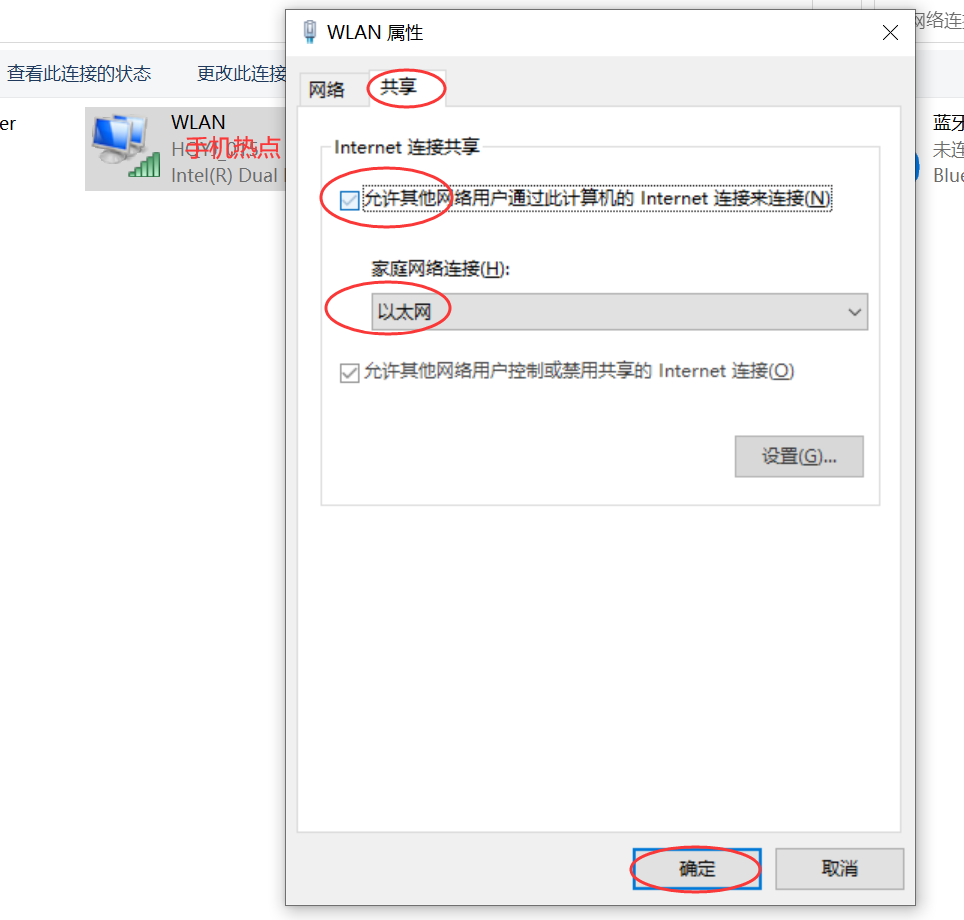
# A9使用以太网卡联网

1. 电脑连接手机热点上网
2. 以太网设置自动获取IP地址和DNS



如果找不到上面这张图片的位置,请参考https://jingyan.baidu.com/article/36d6ed1f3cb6c05ace488335.html

1. 共享WLAN到以太网

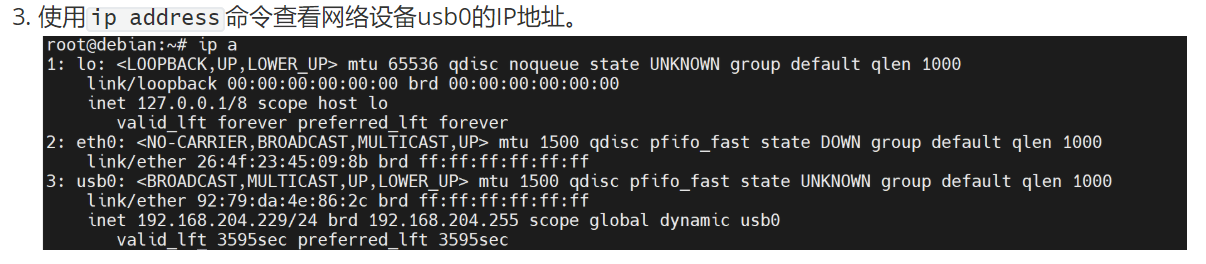


1. 连接网线后，A9单板系统启动时会自动获取IP地址，可以使用ip address命令查看网络设备eth0的IP地址。

注意：如果A9单板分不到IP地址，可能是网线有问题，换根网线试试。

方式2：通过A9通过USB口联网





# 文件IO

1. 文件描述符

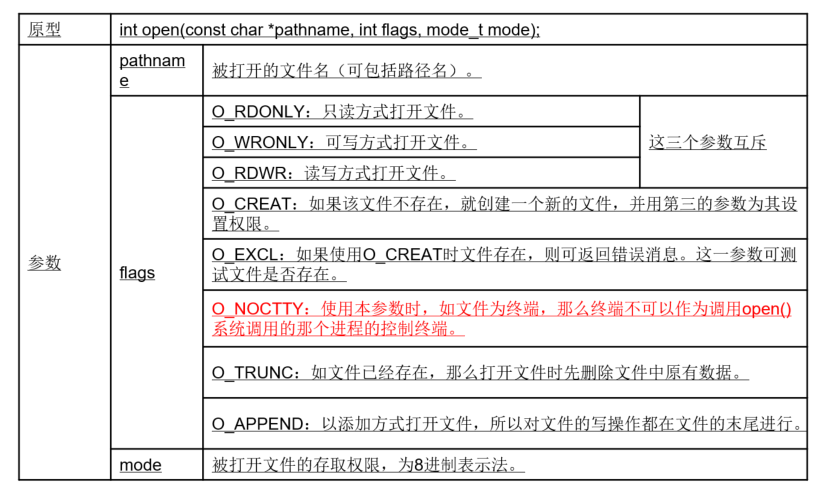
对于内核而言，所有打开文件都由文件描述符引用。

文件描述符是一个非负整数。当打开一个现存文件或创建一个新文件时，内核向进程返回一个文件描述符。

当读、写一个文件时，用open或creat返回的文件描述符标识该文件，将其作为参数传送给read或write。

1. 文件IO系统调用
2. open()

#include <unistd.h>



成功返回打开文件的文件描述符

失败返回-1

1. close()

#include <unistd.h>

调用close()函数可以关闭一个打开的文件。

1. read()

调用read()函数可以从一个已打开的可读文件中读取数据。

#include <unistd.h>

ssize\_t read(int fd, void \*buf, size\_t count);

参数1：打开文件的文件描述符

参数2：缓冲区的地址

参数3：缓冲区的长度

read()调用成功返回读取的字节数，如果返回0，表示到达文件末尾，如果返回-1，表示出错，通过errno设置错误码。

读操作从文件的当前位移量处开始，在成功返回之前，该位移量增加实际读取的字节数。

1. write()

调用write()函数可以向一个已打开的可写文件中写入数据。

#include <unistd.h>

ssize\_t write(int fd, const void \*buf, size\_t count);

参数1：打开文件的文件描述符

参数2：写入数据的缓冲区地址

参数3：将要写入的字节数

write()调用成功返回已写的字节数，失败返回-1，并设置errno。

write()出错的常见原因：磁盘已满或者超过了一个给定进程的文件长度限制。

# OpenCV

1. 什么是OpenCV

OpenCV是一个基于BSD许可（开源）发行的跨平台计算机视觉和机器学习软件库，可以运行在Linux、Windows、Android和Mac OS操作系统上。 它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

1. OpenCV能做什么
2. 调用摄像头驱动采集图像
3. 对图像进行预处理
4. 目标检测（人脸检测、车牌检测等）
5. 安装OpenCV

sudo apt install libopencv-dev

1. OpenCV参考资料

<https://opencv.org/>

1. OpenCV相关知识
2. VideoCapture类

功能：可以实现从摄像机或者视频文件中读取每一帧图像所必须的信息

成员函数：open(int device);

open(string filename);

如果打开的是视频文件，只需要给出视频文件的文件名即可，opencv将会打开对应的视频文件，并准备后序的读取操作，如果打开成功，那么便可以从中获取图像帧，可以通过成员函数isOpened来判断，如果打开成功会返回true,如果文件不存在会返回false，还有如果视频被编码时所使用的编解码器未知，同样会返回false。

如果是使用摄像头来获取视频，需要指明使用哪个摄像头，-1表示默认，0表示第0个，1表示第一个。

1. read从视频流中读取图像

bool read(Mat frame);

一旦有了一个VideoCapture对象，便可以从中读取图像了，该函数从VideoCapture表示的文件中读取下一帧数据，并将数据插入你提供的变量中，同时自动更新VideoCapture对象，以便下次调用read时能够继续返回下一帧图像，如果读完了，则会返回false，同时为函数提供的序列对象也会为空。

1. imshow

void imshow(const string& winname, InputArray mat);

函数功能：用于在指定的窗口中显示图像。如果窗口是用CV\_WINDOW\_AUTOSIZE（默认值）标志创建的，那么显示图像原始大小。否则，将图像进行缩放以适合窗口。而imshow 函数缩放图像，取决于图像的深度：

参数1：要显示的窗口的名字

参数2：InputArray类型的mat，需要显示的图像

1. waitKey

函数格式：waitKey(x);

函数功能：等待键盘事件的函数

参数x ：等待x毫秒，如果在x毫秒期间，按下任意键，则立刻结束并返回按下键的ASCll码，否则返回-1

若 x=0或者一个负数，那么会无限等待下去，直到有按键按下。

1. imwrite

将抓取的图像数据保存成文件。

imwrite(“image.jpg”, image);

参数1：保存成图像文件的文件名。

参数2：摄像头抓取到的图像数据。

1. 需要加的头文件

#include <opencv2/opencv.hpp>

using namespace cv;

1. 如何编译

g++ picShow.cpp -lopencv\_core -lopencv\_highgui -lopencv\_imgcodecs

# C++

C面向过程的语言，C++面向对象的语言

1. C++源文件名.cpp
2. C++完全兼容C语言
3. C++在C的基础上做了扩展，新增了很多特性、关键字、语法、设计思想
4. C++的命名空间

作用：解决命名冲突的问题

namespce 命名空间名

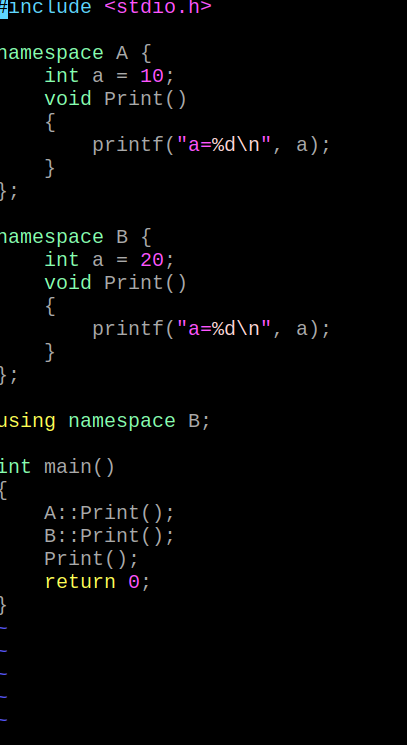
{

变量定义;

函数定义;

类型定义;

};



总结：

使用命名空间中的成员：

1. 命名空间::变量名; (使用域作用域运算符)
2. using namespace 命名空间名;

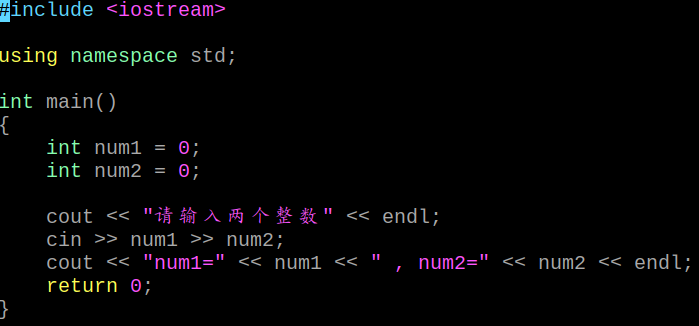
变量名

1. C++的输入输出

cout 标准输出

cin 标准输入

案例1：输入两个整数，打印输出



1. C++的模板

什么是模板：ppt模板、简历模板

泛型编程：广泛的数据类型，算法实现不变，数据类型可变。

算法只需要写一次，把可变的地方抽取成变量。

模板实现两数求和

T add(T a, T b)

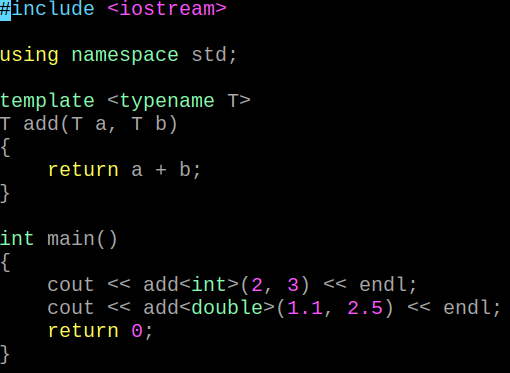
{

Return a + b;

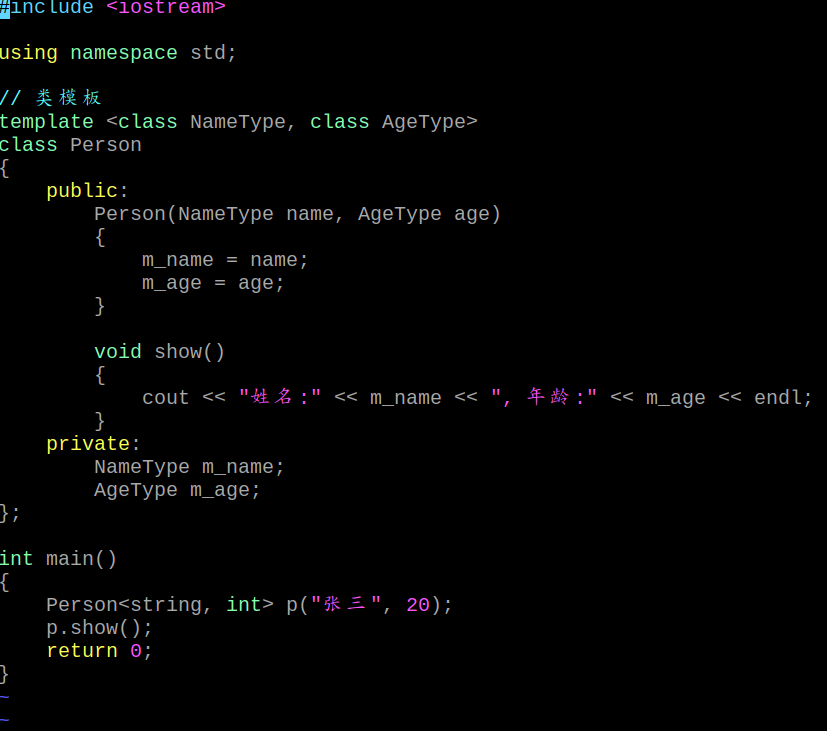
}

模板分为两类：函数模板和类模板

1. 函数模板



1. 类模板



1. string类型

头文件：#include <string>

命名空间：using namespace std;

1. string部分构造函数

构造一个空的string对象：string s;

用C语言的字符串构造string对象：string s(“hello world”);

1. string字符串拼接

使用+拼接多个string对象

string s1(“hello”);

string s2(“world”);

string s3 = s1 + s2;

使用+拼接至少一个string对象和多个C风格的字符串

string s = “hello” + string(“world”);

1. string字符串比较

直接使用关系运算符（==、!=、>、>=、<、<=）比较

1. 标准模板库STL

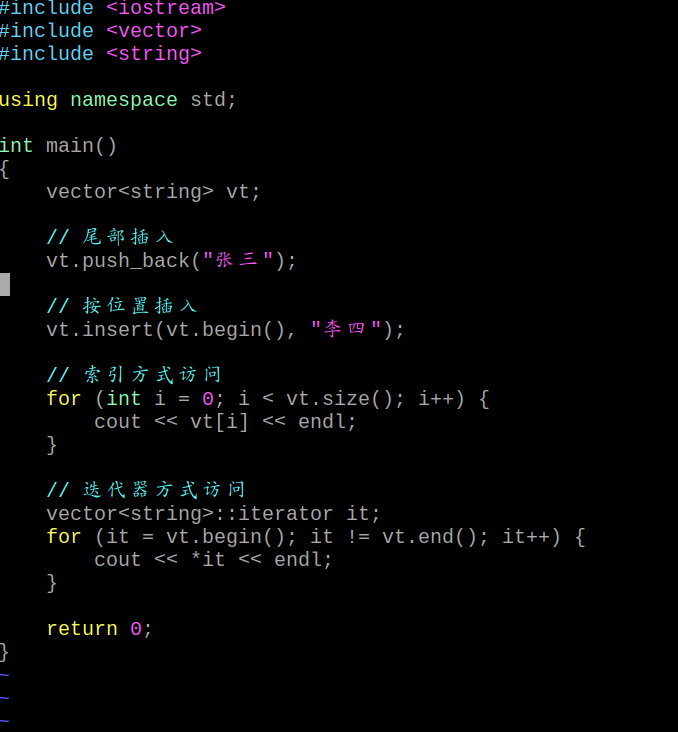
STL：标准模板库，标准模板库里实现的容器类都是用C++的模板实现的。

1. vector容器

#include <vector>

C语言的数组是固定长度的，存在问题：资源要么浪费，要么不够用。

vector是动态数组，当容器的容量用尽的时候，会自动申请一块更大的内存，然后将原来的内存空间中的数据拷贝到新申请的内存空间里，再释放掉原来的内存空间。



1. map容器

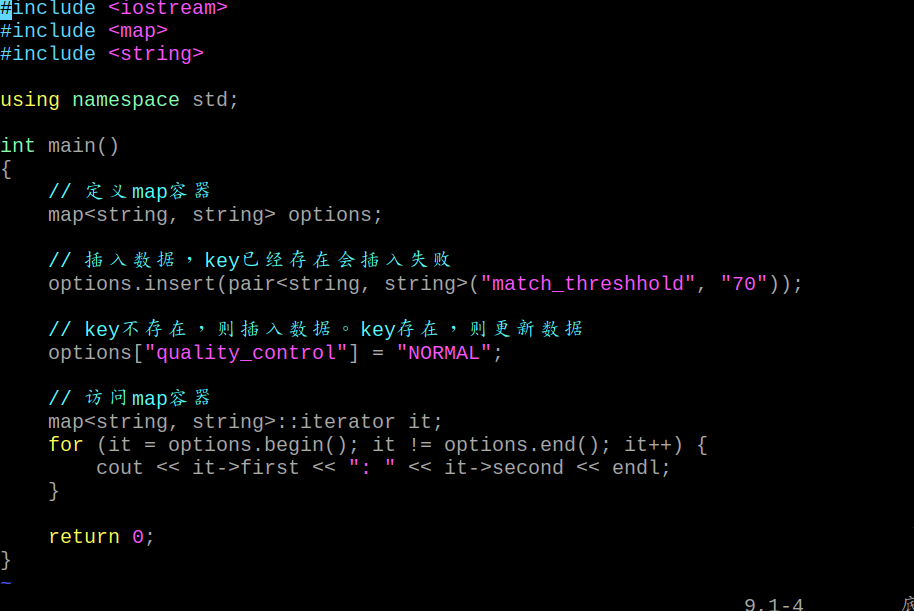
文件件：#include<map>

底层数据类型：红黑树实现的

Map是key-value的键值对，可以根据key找到对应的value，key和value是一一对应的。

Map容器的key是不允许重复的。

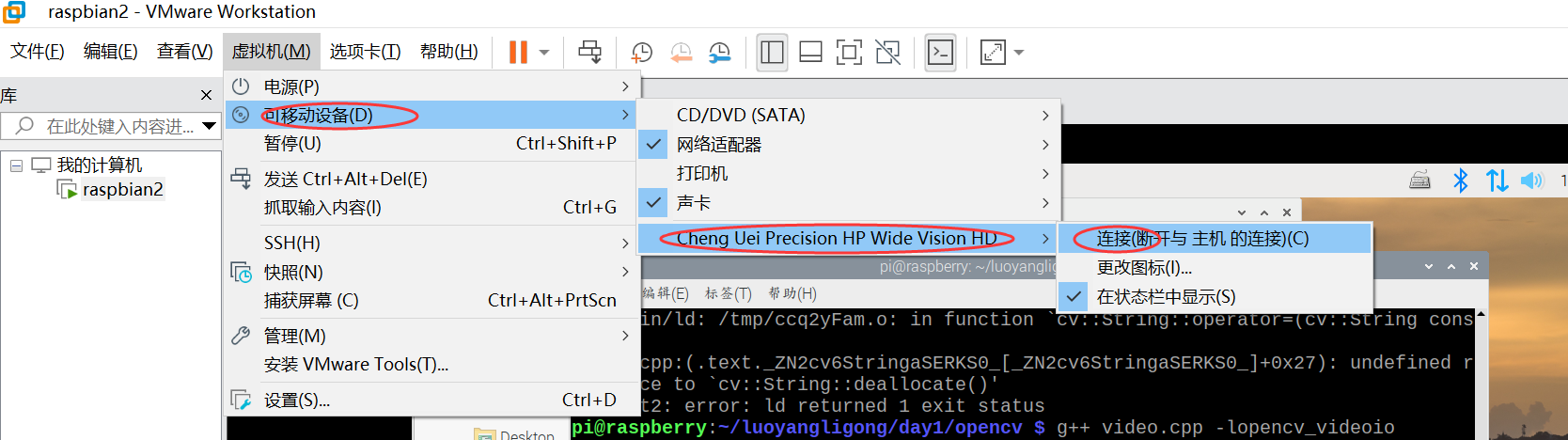
Map容器的key会自动排序，排序的方式是根据key值的大小，从小到大排序。

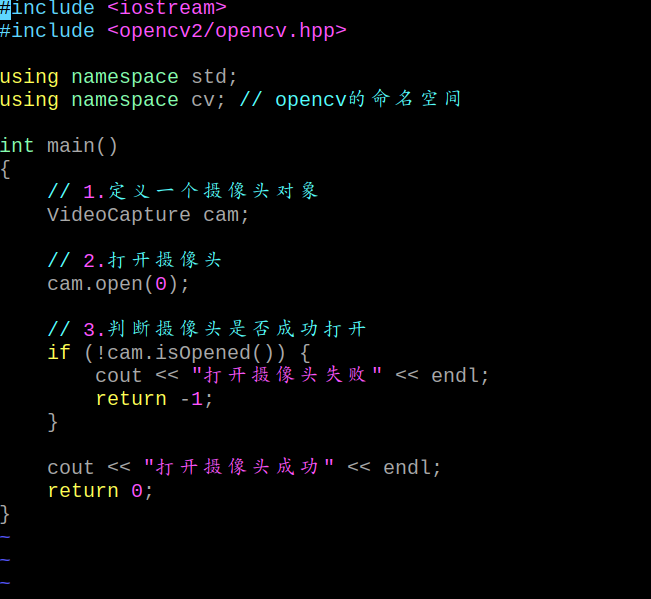


# 视频采集

1. 打开摄像头

如果打开摄像头失败，参照下图连接摄像头





1. 采集图像

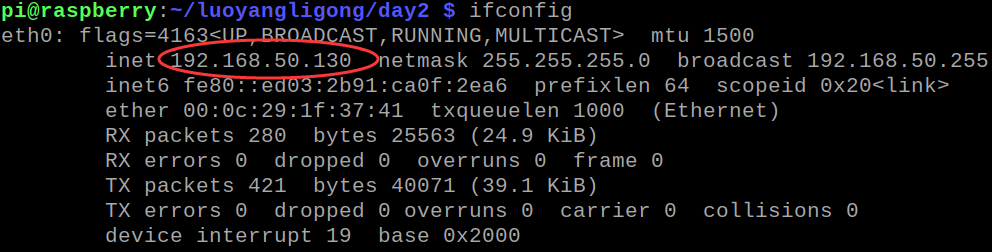


Mat image;

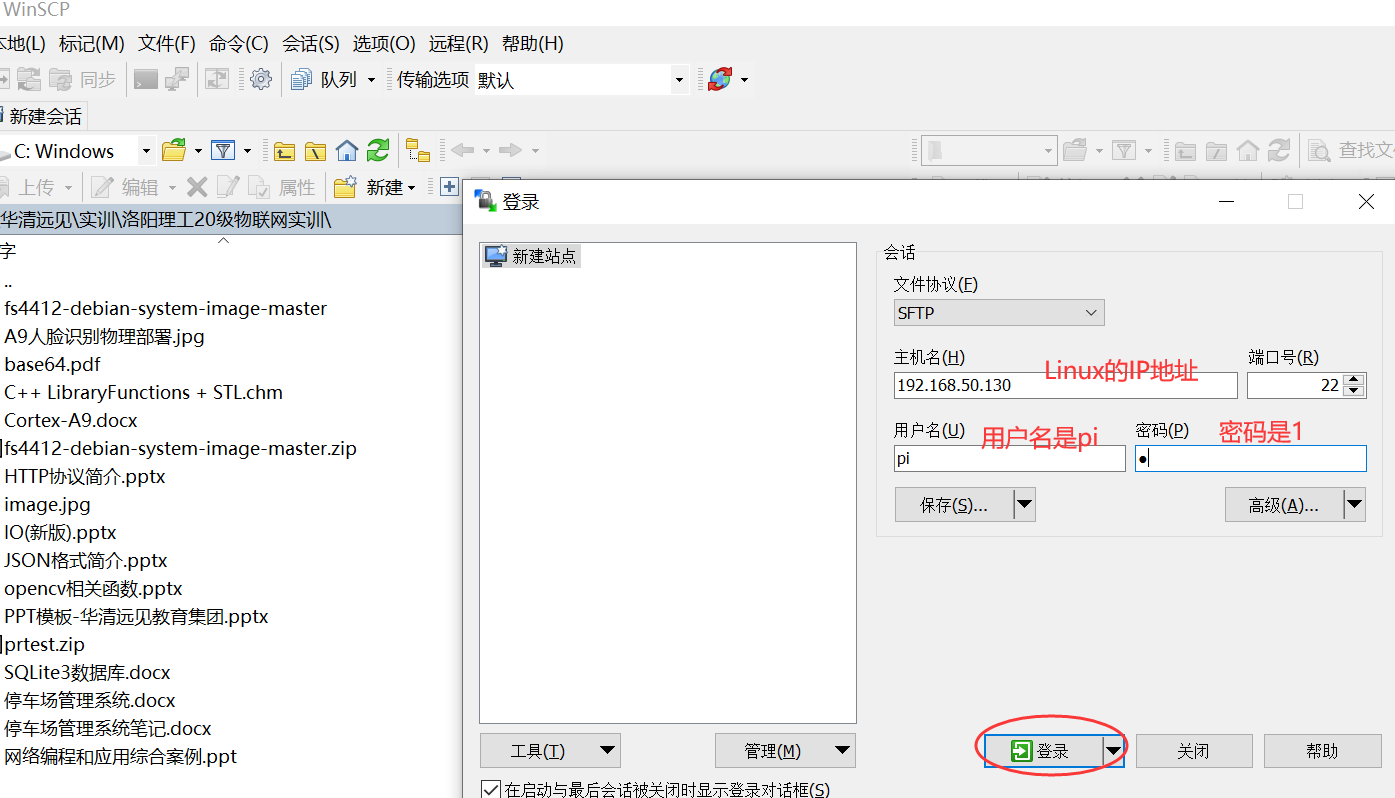
imwrite(“image.jpg”, image);

# WinScp用于Windows和Linux互传文件

1. 安装WinScp
2. 查询Linux的IP地址

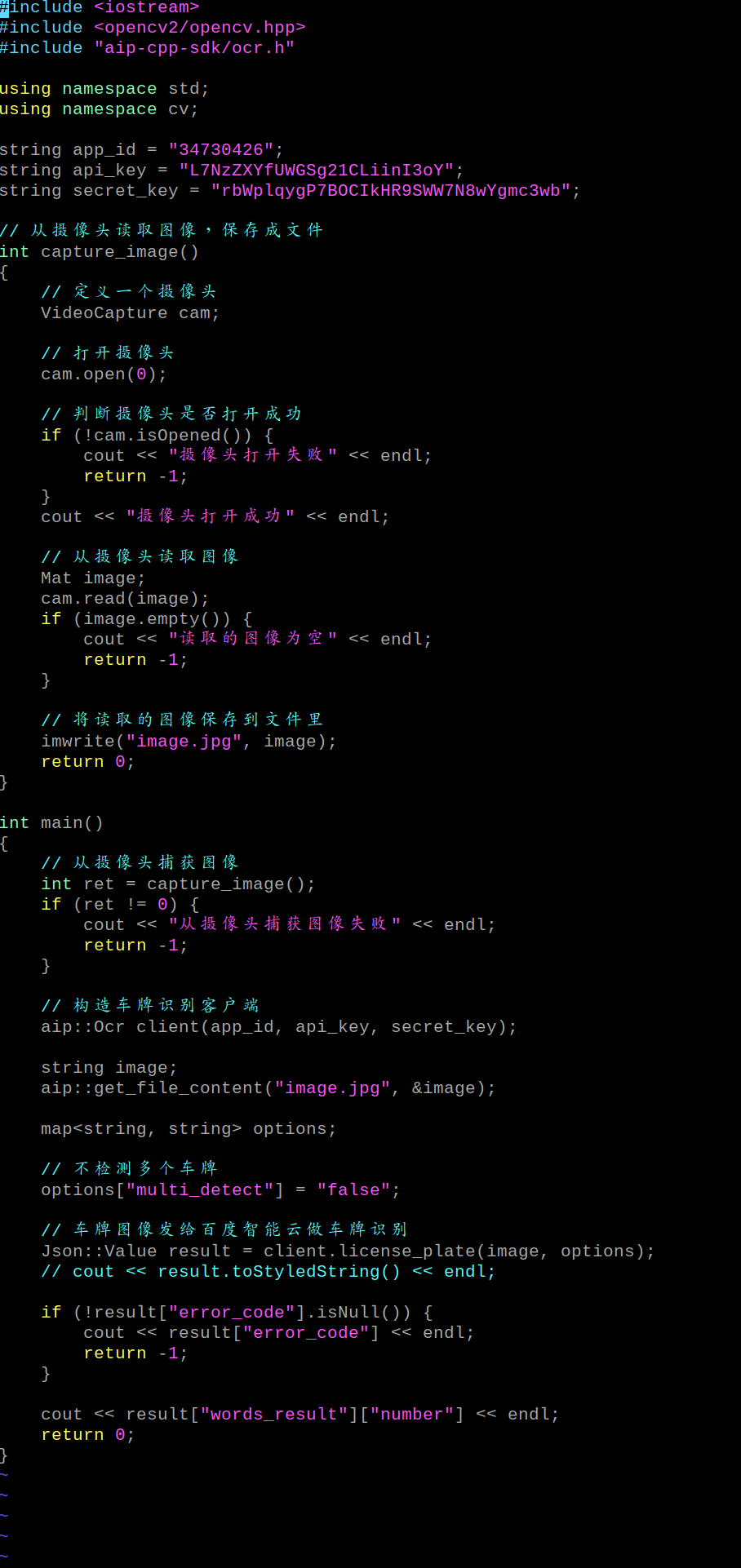


1. 打开WinScp，连接Linux



# 车牌识别

1. 注册百度智能云，领取免费车牌识别资源，创建车牌识别应用。
2. 将百度智能云的开发包（aip-cpp-sdk）放到出牌识别的开始目录里。
3. 实现车牌识别。



1. 编译车牌识别程序



# Makefile

1. make

工程管理器

make会在当前目录下找一个叫Makefile的一个文件（M可以大写，也可以消息），然后按照Makefile里定义规则编译工程代码。

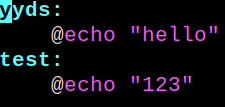
1. Makefile的语法规则

目标：依赖

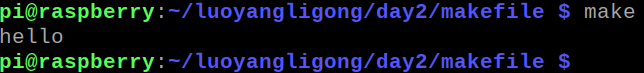
（tab键）命令

一个Make file可以有多个目标。

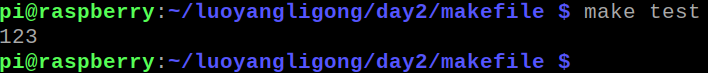
1. 多目标，目标之间没有依赖关系



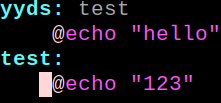
调用make时，如果不指定目标，默认执行第1个目标。

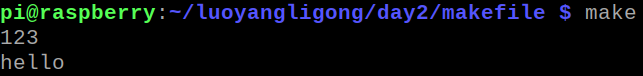


要执行指定的目标，可以在make后面加上目标。



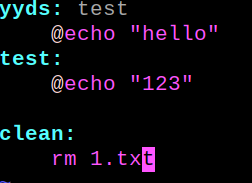
1. 多目标，目标之间有依赖关系

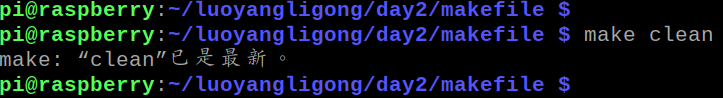




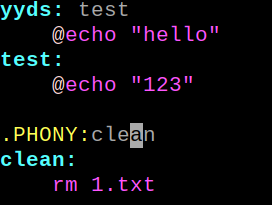
执行make，在执行yyds对应的命令前，先执行了test对应的命令。

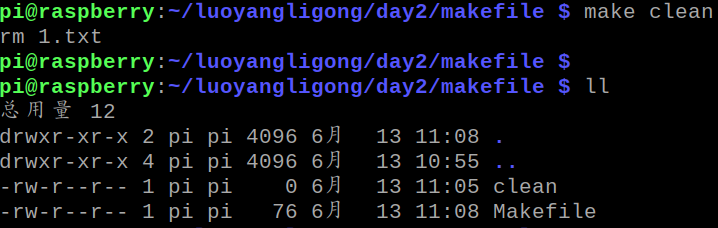
1. 伪目标（假目标）





定义了伪目标之后：





1. 自定义变量

可以定义一个OBJS变量，用于存放所有的目标文件。

OBJS=目标文件

1. 预定义变量

CXXFLAGS：C++编译器的编译选项

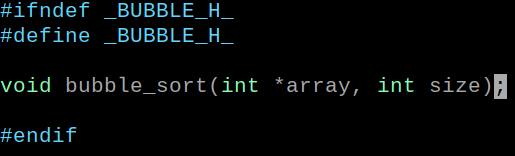
1. 自动变量

$@：代表目标

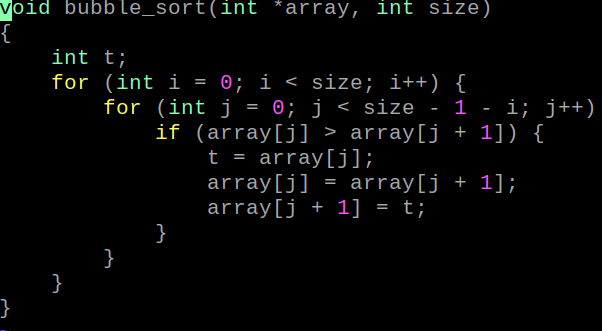
$^：代表依赖

案例：多文件编译，一个文件实现冒泡排序函数，另一个文件调用冒泡排序函数。

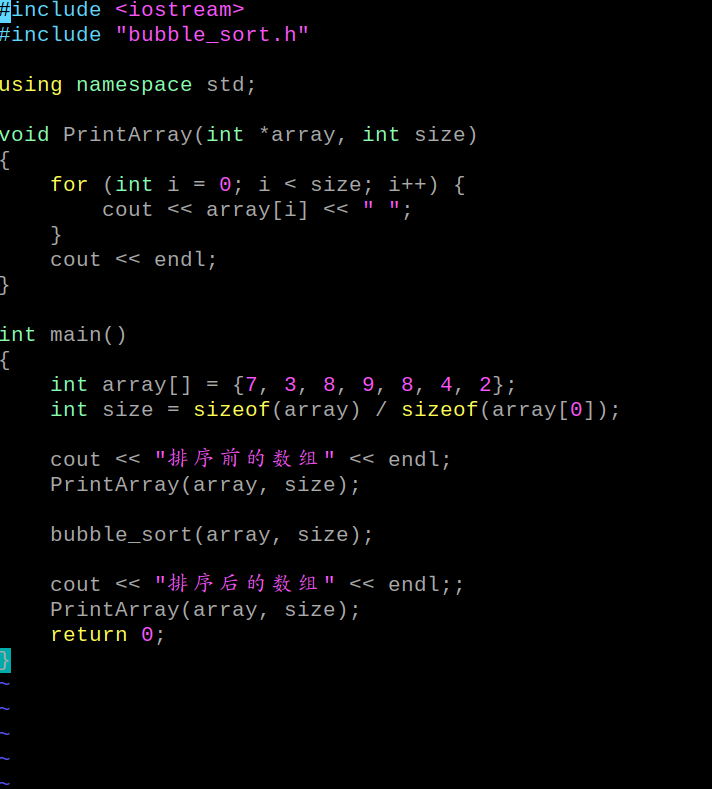
bubble\_sort.h



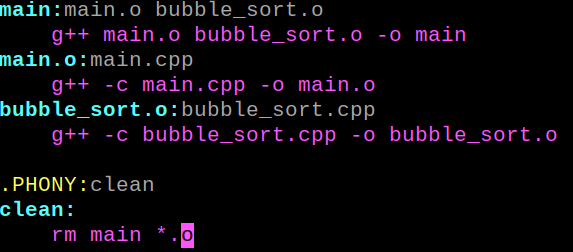
bubble\_sort.cpp

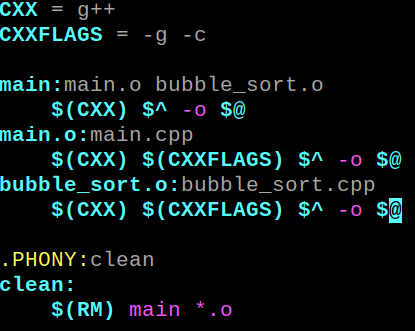


main.cpp



Makefile





# 读取和控制LED灯的状态

LED2是心跳指示灯，CPU工作期间会定时闪烁。

LED3~LED5可以同程序控制，可以通过sysfs控制和读取LED灯的状态。

查看LED3的状态：



打开LED3（点灯）：

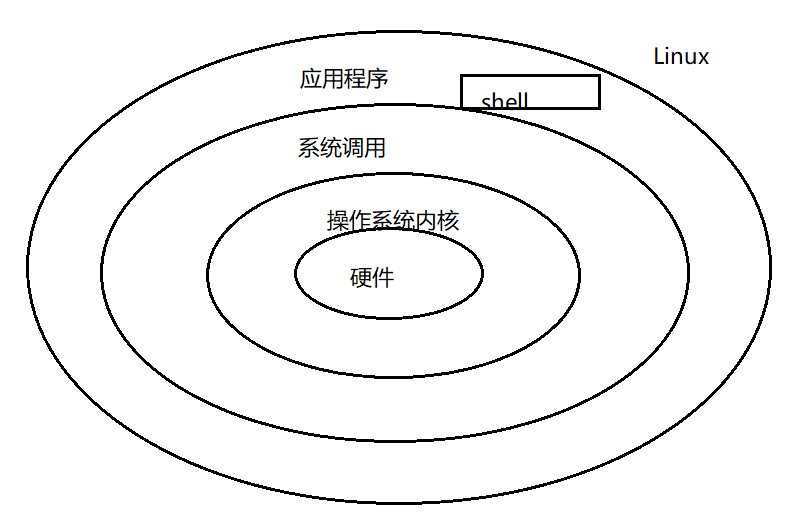


关闭LED3（熄灯）：



练习：用文件IO的函数实现打开LED3和关闭LED3。

# 如何使用Linux



1. Vim编辑器

: 打开文件默认进入 i, a, o

命令模式 🡨--------------普通模式-------------------🡪编辑模式（插入模式）

🡨----------- esc

命令模式下的部分命令：

w（保存，不退出）

q（退出，不保存）

wq（保存，退出）

set number/set nu（显示行号）

set nonu（取消显示行号）

# 控制蜂鸣器

可以通过程序控制底板上的蜂鸣器：

5 struct input\_event {

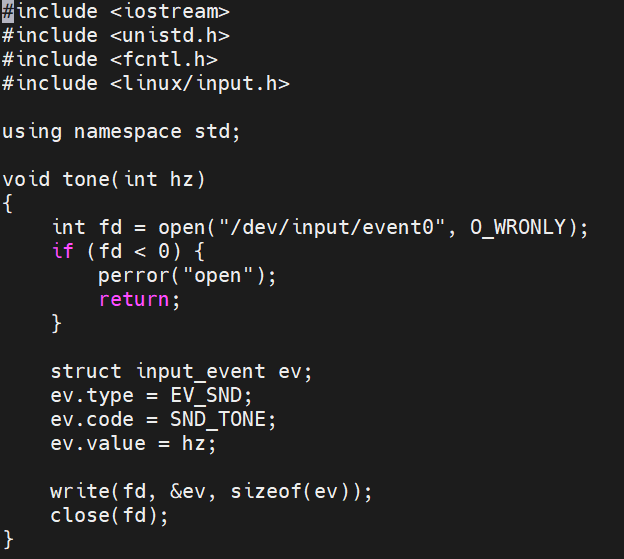
6 struct timeval time; // 表示输入时的时间

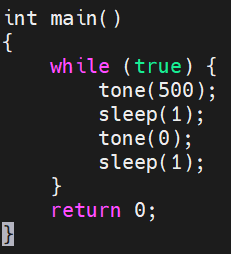
7 \_\_u16 type; // 表示输入的设备，比如键盘、鼠标等

8 \_\_u16 code; // 根据不同的type有code，比如键盘类型，按下的哪个按键

9 \_\_s32 value; // 根据不同的type和code决定，比如按键A的按下和松开

10 };



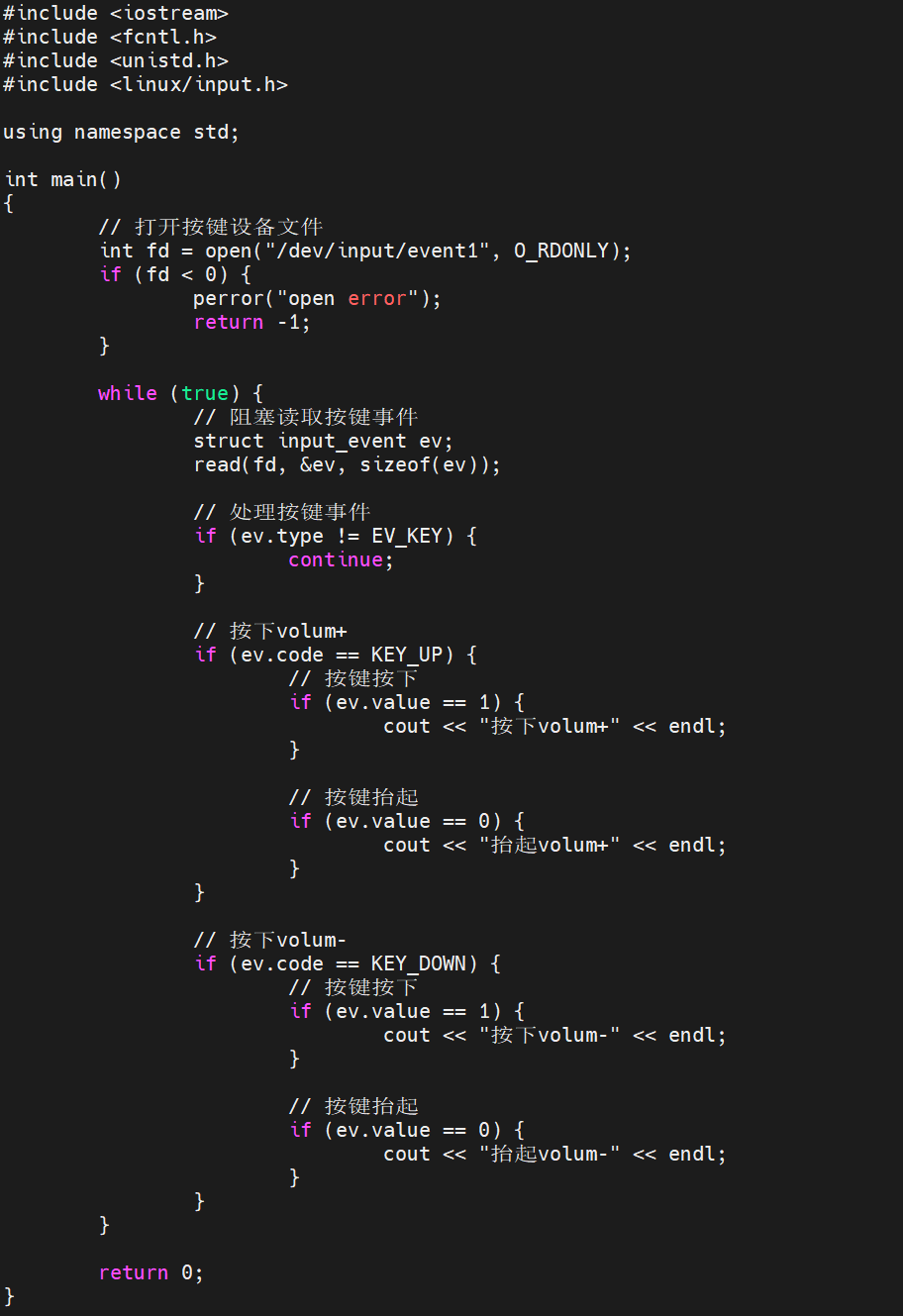


参数hz是蜂鸣器的声音频率，设置为0表示停止蜂鸣器。

# 读取按键状态

底板上有3个用户按键，power按键，用于关机。音量+-按键的状态可以通过程序读取。

在一个输入设备中，每发生一次输入事件，struct input\_event就会被填充。

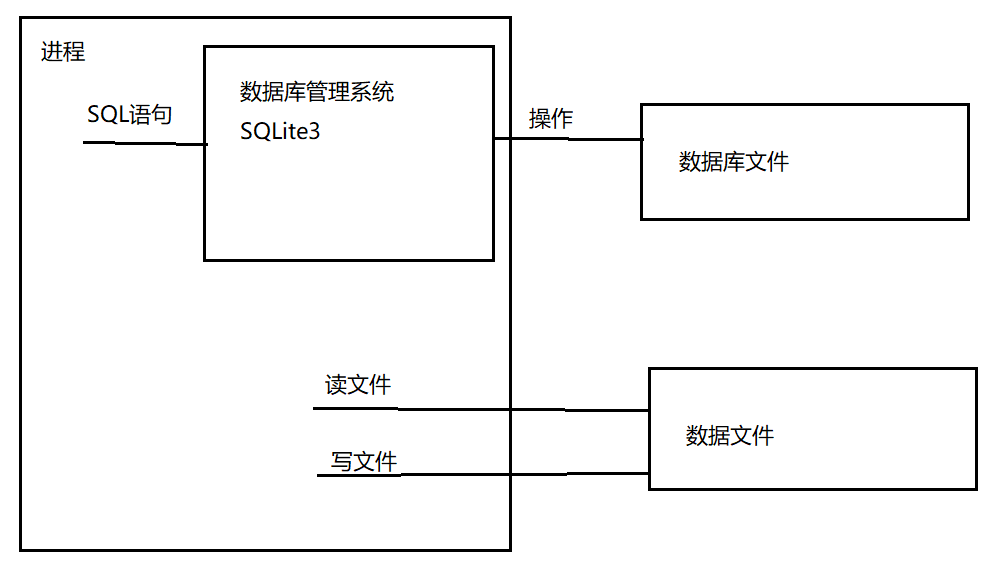


g++ license\_plate.cpp -I/usr/include/opencv4 -I/usr/include/jsoncpp -lopencv\_videoio -lopencv\_core -lopencv\_imgcodecs -lcurl -ljsoncpp -lcrypto

# 数据库SQLite3

数据：能够输入计算机并能被计算机程序识别和处理的信息集合。

数据库：数据库是在数据库管理系统管理和控制之下，存放在存储介质上的数据集合。



SQLite的源代码是C，其源代码完全开放。SQLite第一个Alpha版本诞生于2000年5月。 他是一个轻量级的嵌入式数据库。

SQLite有以下特性：

零配置一无需安装和管理配置；

储存在单一磁盘文件中的一个完整的数据库；

数据库文件可以在不同字节顺序的机器间自由共享；

支持数据库大小至2TB；

足够小，全部源码大致3万行c代码，250KB；

比目前流行的大多数数据库对数据的操作要快；

1. 安装SQLite3

树莓派安装方式：



A9单板安装方式：

Usb连接手机热点上网

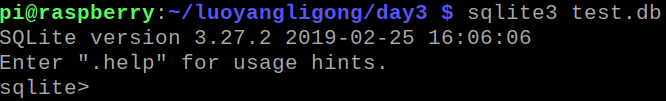
apt install sqlite3

apt install libsqlite3-dev

1. SQLite3的系统命令

SQLite3的系统命令是以点开头的，又叫点命令。

打开数据库，数据库如果不存在会创建数据库。



查看数据库文件存放的位置



查看数据库里的所有表

.tables

查看表的结构

.schema

退出数据库

.quit

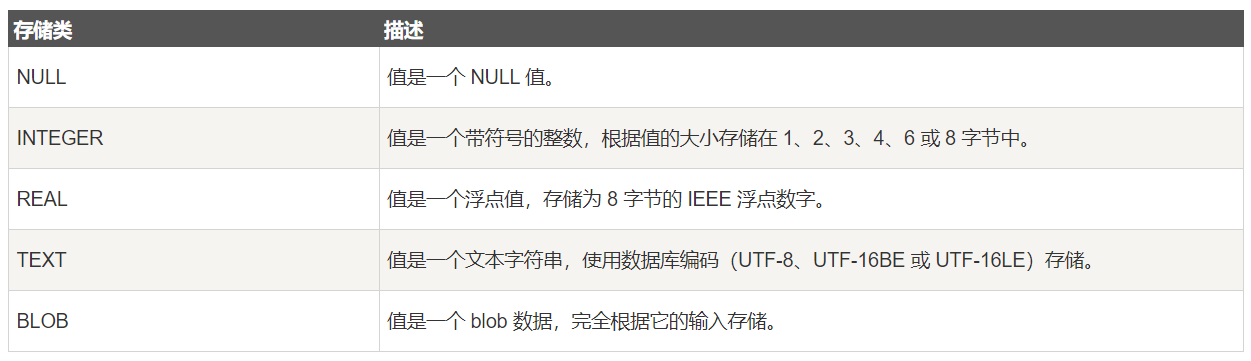
.exit

1. SQLite的网站

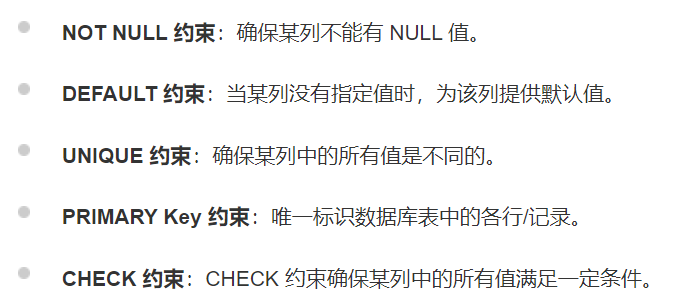
官网：<https://www.sqlite.org/index.html>

菜鸟教程：<https://www.runoob.com/sqlite/sqlite-c-cpp.html>

1. SQLite的数据类型



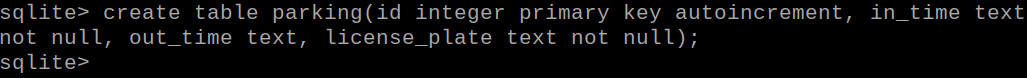
1. SQLite的约束



1. SQL语句

SQL语句不以点开头，要以分号结尾。

1. 创建表



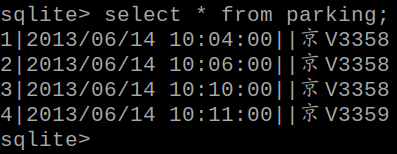
1. 删除表

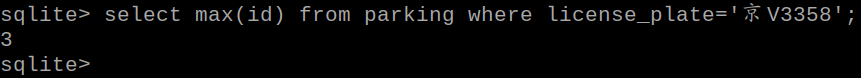


1. 向表中添加新记录



1. 查询表中的记录





1. 更新表中的记录



1. SQLite3的编程接口

int sqlite3\_open(char \*path, sqlite3 \*\*db);

功能：打开sqlite数据库

path： 数据库文件路径

db： 指向sqlite句柄的指针

返回值：成功返回0，失败返回错误码(非零值)

int sqlite3\_close(sqlite3 \*db);

功能：关闭sqlite数据库

返回值：成功返回0，失败返回错误码

const char \*sqlite3\_errmsg(sqlite3 \*db);

返回值：返回错误信息

int sqlite3\_exec(sqlite3 \*db, const char \*sql, sqlite3\_callback callback, void \*arg, char \*\*errmsg);

功能：执行SQL操作

db：数据库句柄

sql：SQL语句

callback：回调函数

arg：传递给callback的参数

errmsg：错误信息指针的地址

返回值：成功返回0，失败返回错误码

typedef int (\*sqlite3\_callback)(void \*para, int f\_num, char \*\*f\_value, char \*\*f\_name);

功能：每找到一条记录自动执行一次回调函数

para：传递给回调函数的参数

f\_num：记录中包含的字段数目

f\_value：包含每个字段值的指针数

f\_name：包含每个字段名称的指针数组

返回值：成功返回0，失败返回-1

char \*f\_value[4];

f\_value[0] -> “1”

f\_value[1] -> “2023/06/14 10:10:00”

f\_value[2] -> “2023/06/14 11:10:00”

f\_value[3] -> “京V3358”

void sqlite3\_free(char \*errmsg);

功能：释放错误信息

参数1：错误信息的指针

1. SQLite3例子

停车场管理系统

表设计：

parking

字段设计：

id：车辆入场标识，主键

in\_time：入场时间

out\_time：出场时间

license\_plate：车牌号

功能设计：

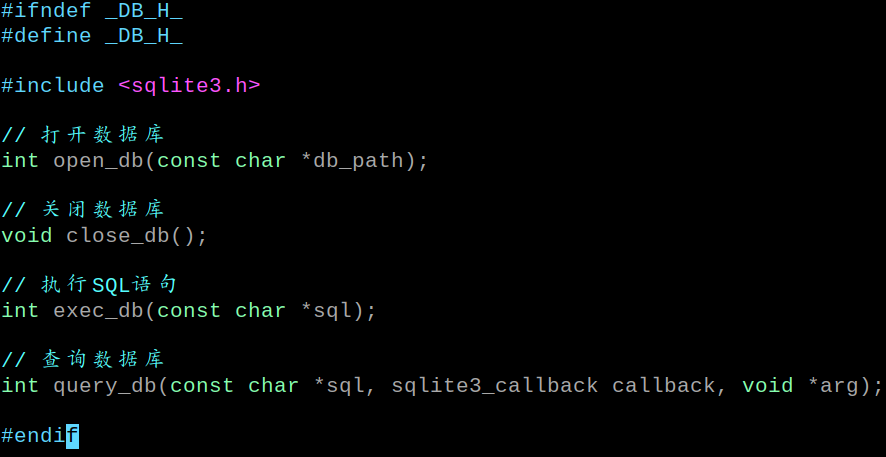
插入车辆信息

// 删除车辆信息 🡪 不支持

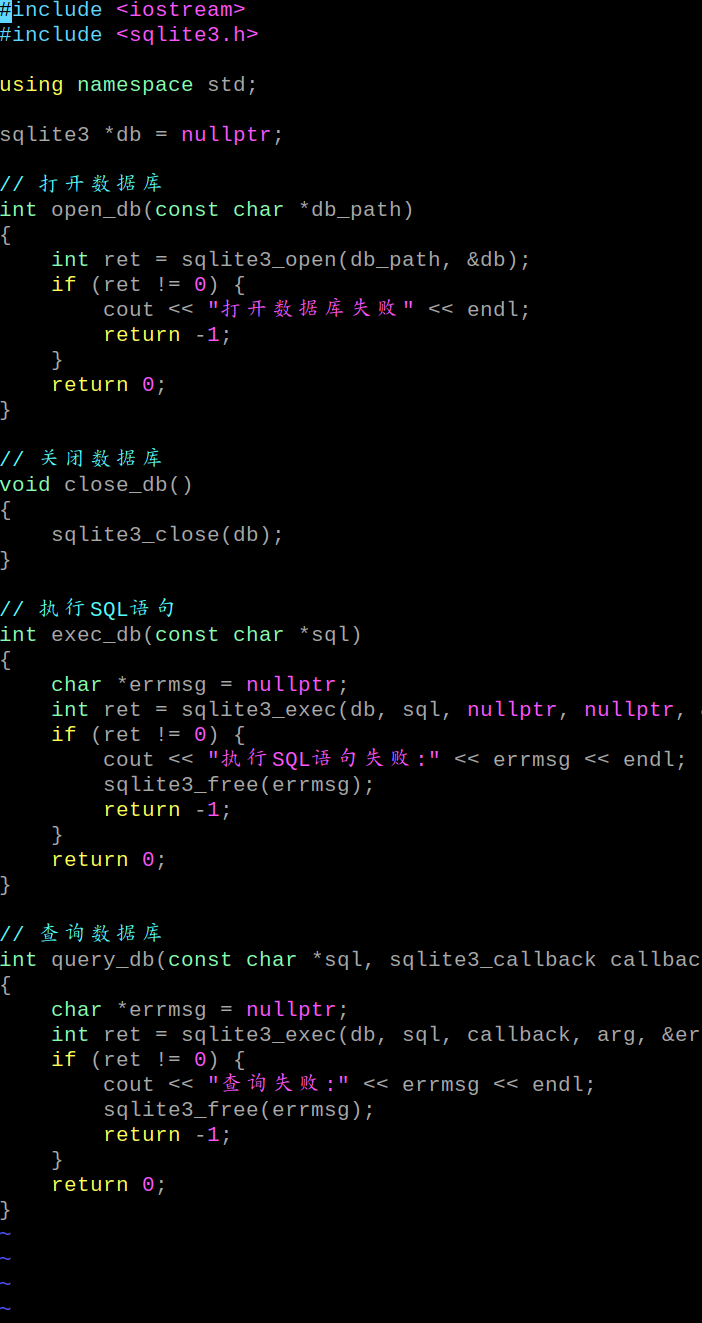
修改车辆信息 🡪 修改出场时间

查询车辆信息

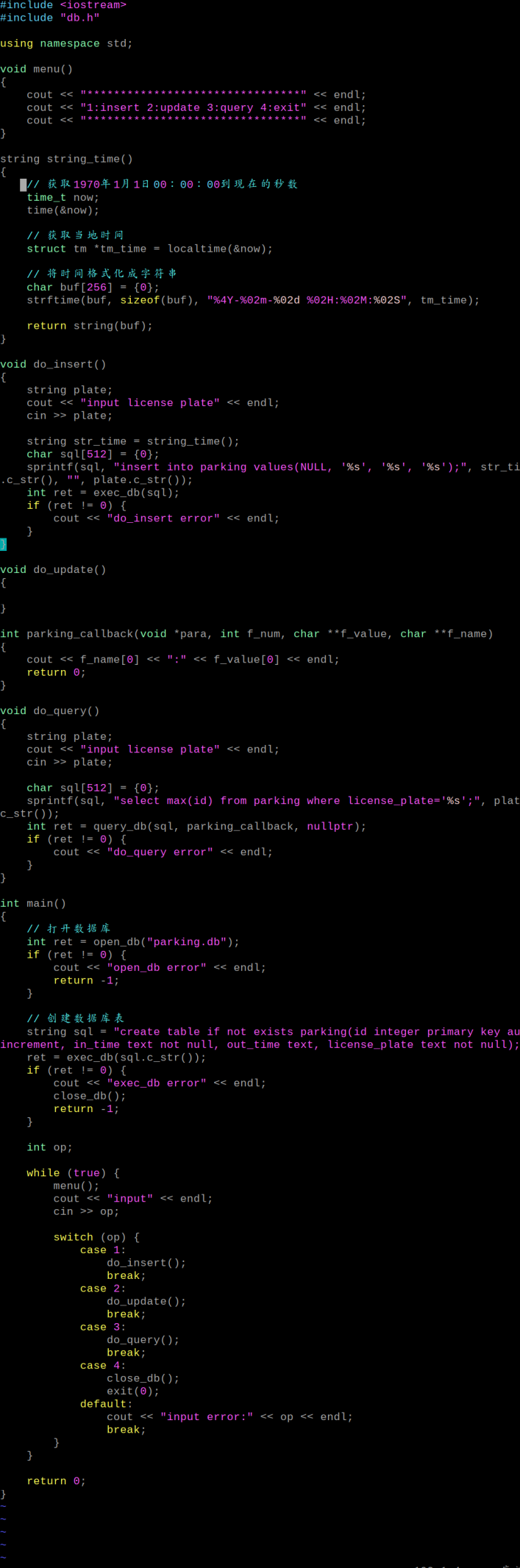
db.h



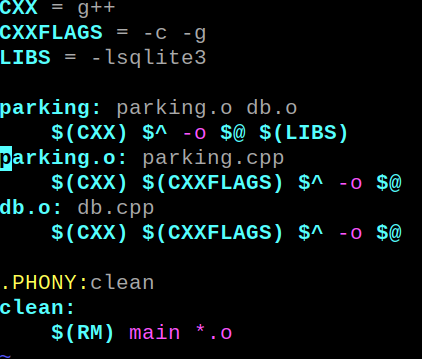
db.cpp



parking.cpp



Makefile



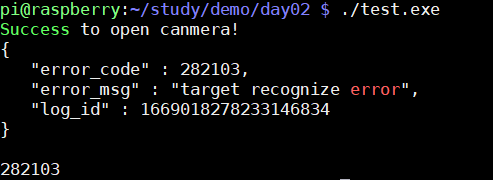
# 第四天的项目介绍

停车场管理系统的模块划分：

1. 进场、出场处理//insert,update
2. 车牌识别//Camer
3. 数据库存储车辆进场、出场信息

停车场管理系统的功能：

1. 车辆进入停车场时，用点灯模拟抬杆操作；用熄灯模拟落杆操作。//insert前点灯,insert后熄灯
2. 进场失败场景，比如车牌识别失败，蜂鸣器响2~3秒。



如果失败的返回的结果是282103

1. 车辆进入停车场时，需要将进场时间、车牌号存入数据库。//ok
2. 车辆离开停车场时，需要将车辆出场时间更新到数据库里。//ok
3. 车辆离开停车场时，需要计算车辆在停车场内停留的时间。//ok,update收费
4. 用volum+按键模拟检测到车辆准备进入停车场；volum+压下表示感应到车辆已经在停车场入口了，volum+抬起表示感应到车辆已经从停车场入口进入停车场了。//insert之后,完成后,管理员按下volum+按键,等待10秒后,volum+自动抬起
5. 用volum-按键模拟检测到车辆准备离开停车场；volum-压下表示感应到车辆已经在停车场出口了，volum-抬起表示感应到车辆已经从停车场出口离开停车场了。//update收费之后,管理员按下volum-按键,等待10秒后,volum-自动抬起
6. 在项目当前目录下新建in和out两个文件夹，in文件夹用于保存车辆进入停车场时抓取的车牌识别图像，用当前时间命名。out文件夹用于保存车辆离开停车场时抓取的车牌识别图像，用当前时间命名。//进入和离开都要拍照,进的时候调用先拍照,然后insert,然后将图片放入in文件;离开的时候,也是先拍照再调用update,然后将图片放入out文件
7. 项目用Makefile实现。

