

# 嵌入式入门知识了解



# 目录

Contents

01 Ubuntu+Windows双系统

02 常规使用与ROS安装

03 CMakeLists编写



# PART 01

## Ubuntu+Windows双系统

在上课之前，  
应该准备好  
Ubuntu和  
windows的双  
系统，并且都  
能正常使用。

### Download Ubuntu Desktop

#### Ubuntu 14.04.1 LTS

The Long Term Support (LTS) version of the Ubuntu operating system for desktop PCs and laptops, Ubuntu 14.04.1 LTS comes with five years of security and maintenance updates, guaranteed.

**Recommended for most users.**

[Ubuntu 14.04.1 LTS release notes](#) 

Choose your flavour

64-bit — recommended ▼

Download

[Alternative downloads and torrents >](#)



# PART 02

## 常规使用

打开终端：Ctrl+Alt+T

### 目录与文件相关命令

pwd：用于显示当前位置的绝对路径。格式：pwd，默认用户根目录为/home/你的用户名。

ls：用于显示当前目录下的文件，默认只显示非隐藏文件。格式：ls。

cd：用于改变你的工作目录。格式：cd 工作目录（一般和ls一起用，tab键补齐），和cd /命令是进入根目录；cd ..命令是进入上一层目录；cd -命令是回到之前的目录输入。

cp：复制命令，格式：cp 文件1路径 文件2路径，和复制命令格式cp -r 目录1路径 目录2路径

rm：删除命令，格式：rm 文件路径，删除目录时使用 rm -r 目录路径



# PART 02

## 常规使用

mdir：删除空目录，非空目录使用rm -r格式

mkdir：在当前目录下创建新目录，格式：mkdir 文件夹名

man：有关命令帮助，查找某个命令的详细用法。格式：man 某个命令。ubuntu中文版man手册配置方法：man默认是英文的，但ubuntu的源里也有中文版的。以下是配置方法。终端输入sudo apt-get install manpages-zh，安装后修改配置文件sudo gedit /etc/manpath.config，将所有的/usr/share/man替换为/usr/share/man/zh\_CN，保存即可。

Tab

自动补全



# PART 02

## ROS介绍

通常这样解释ROS：

1. 通道：ROS提供了一种发布-订阅式的通信框架用以简单、快速地构建分布式计算系。
2. 工具：ROS提供了大量的工具组合用以配置、启动、自检、调试、可视化、登录、测试、终止分布式计算系统。
3. 强大的库：ROS提供了广泛的库文件实现以机动性、操作控制、感知为主的机器人功能。
4. 生态系统：ROS的支持与发展依托着一个强大的社区。ros.org尤其关注兼容性和支持文档，提供了一套“一站式”的方案使得用户得以搜索并学习来自全球开发者数以千计的ROS程序包。



# PART 02

## ROS介绍

摘自《ROS by Example》的解释：

ROS的首要目标是提供一套统一的开源程序框架，用以在多样化的现实世界与仿真环境中实现对机器人的控制。

\* 关于ROS的“通道（plumbing）”（摘自《ROS by Example》

ROS的核心是节点（node）。节点是一小段用Python或C++写成的程序，用来执行某个相对简单的任务或进程。多个节点之间互相传递信息（message），并可以独立控制启动或终止。某一节点可以面向其它节点针对特定标题（topic）发布信息或提供服务（service）。

例如：现有以节点将传感器读数传递至机器人控制器，在“/head\_sonar标题”下存在一条信息包含有变量值“0.5”，即意味着传感器检测到的当前物体距离为0.5米。任何一个想要知道该传感器读数的节点都只要订阅（subscribe）/head\_sonar标题即可。为了便于使用该读数，针对该订阅者的节点会定义一个回调函数，每当有新的信息传递到订阅者标题时，即执行该函数。上述流程的运行频率取决于发布者节点（publisher node）更新信息的频率。

此外，节点还可以用来定义一个或多个服务（service）。ROS中服务的作用是在接收到来自其它节点的请求时回复该节点或执行某项任务。例如：控制LED灯的开关是一个服务；移动机器人在给定起始和目标位置的前提下返回导航路线规划也是一个服务。



# PART 02

## ROS指令

创建 ROS 工作空间

启动 ROS

```
$ roscore
```

创建工作环境

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

```
$ catkin_init_workspace
```

编译 ROS 程序

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

添加程序包到全局路径

```
$ echo "source catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
$ source ~/.bashrc
```



# PART 02

## ROS指令

Package 相关操作

创建 Package 并编译

```
$ cd ~/catkin_ws/src
```

```
$ catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

查找 Package

```
$ rospack find [package name]
```

查看 Package 依赖

```
$ rospack depends <package_name>
```

```
$ rospack depends1 <package_name>
```



# PART 02

## ROS指令

Node 相关操作

查看所有正在运行的 Node

\$ rosnode list

查看某节点信息

\$ rosnode info [node\_name]

运行 Node

\$ rosrun [package\_name] [node\_name] [\_\_name:=new\_name]



# PART 02

## ROS指令

Topic 相关操作

查看 rostopic 所有操作

```
$ rostopic -h
```

查看所有 Topic 列表

```
$ rostopic list
```

图形化显示 topic

```
$ rosrun rqt_graph rqt_graph
```

```
$ rosrun rqt_plot rqt_plot
```

查看某个 Topic 信息

```
$ rostopic echo [topic]
```

查看 Topic 消息格式

```
$ rostopic type [topic]
```

```
$ rosmmsg show [msg_type]
```

向topic发布消息

```
$ rostopic pub [-1] <topic> <msg_type> [-r 1] -- [args] [args]
```



# PART 02

## ROS指令

Service 相关操作

查看所以service操作

```
$ rosservice -h
```

查看 service 列表

```
$ rosservice list
```

调用 service

```
$ rosservice call [service] [args]
```

查看 service 格式并显示数据

```
$ rosservice type [service] | rossrv show
```

设置service parameter

```
$ rosparam set [parame_name] [args] + rosservice call clear
```

获得parameter

```
$ rosparam get [parame_name]
```

加载parameter

```
$ rosparam load [file_name] [namespace]
```

删除parameter

```
$ rosparam delete
```



# PART 02

## ROS指令

ag 相关操作

录制所有topic变化

```
$ rosbag record -a
```

记录某些topic

```
$ rosbag record -O subset <topic1> <topic2>
```

查看bag信息

```
$ rosbag info <bagfile_name>
```

回放

```
$ rosbag play (-r 2) <bagfile_name>
```





# PART 03

---

CMakeLists.txt

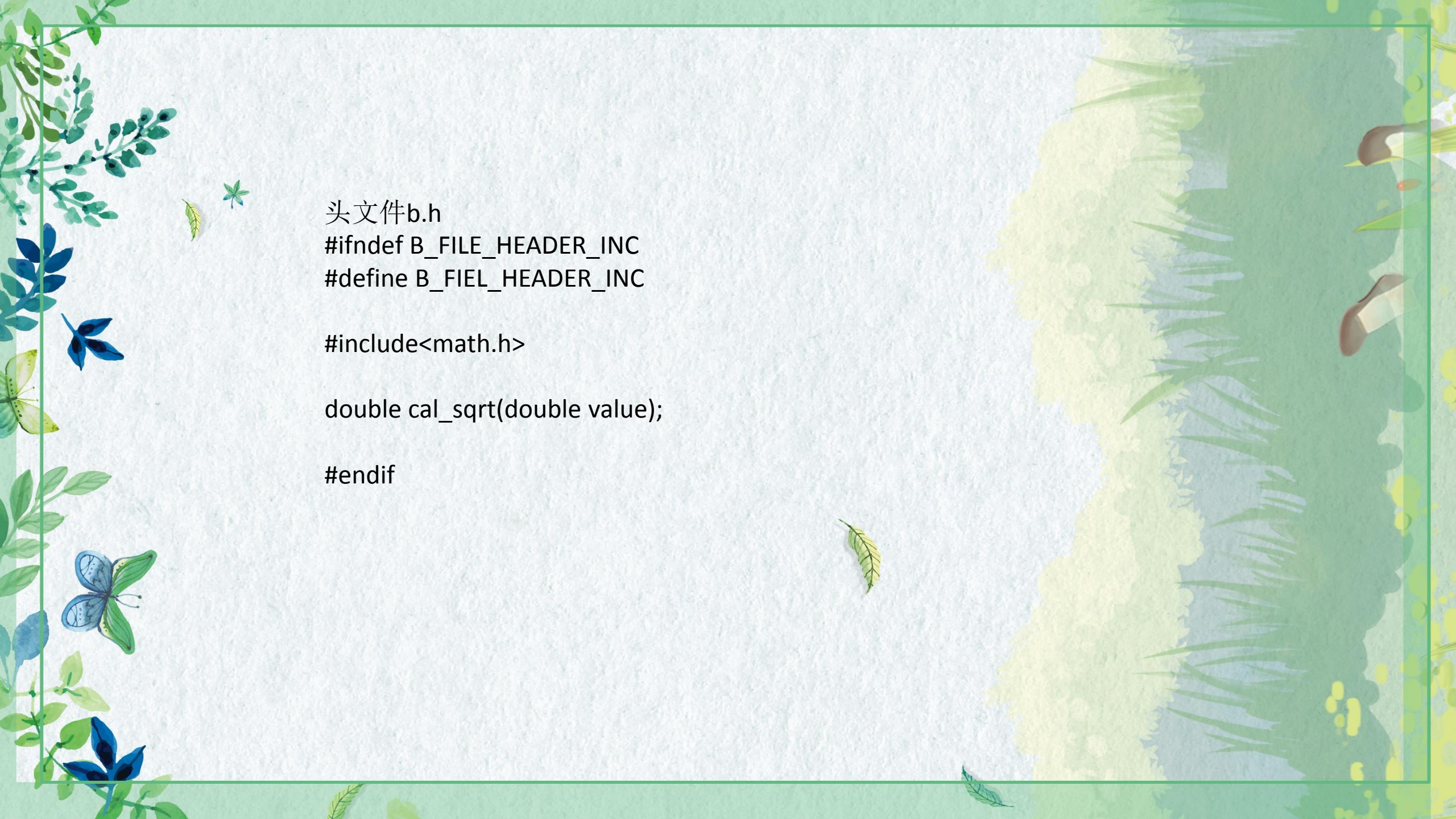
---



准备如下的文件结构图：

```
.
├── build
├── CMakeLists.txt
├── include
│   └── b.h
└── src
    ├── b.c
    └── main.c
```





头文件b.h

```
#ifndef B_FILE_HEADER_INC
```

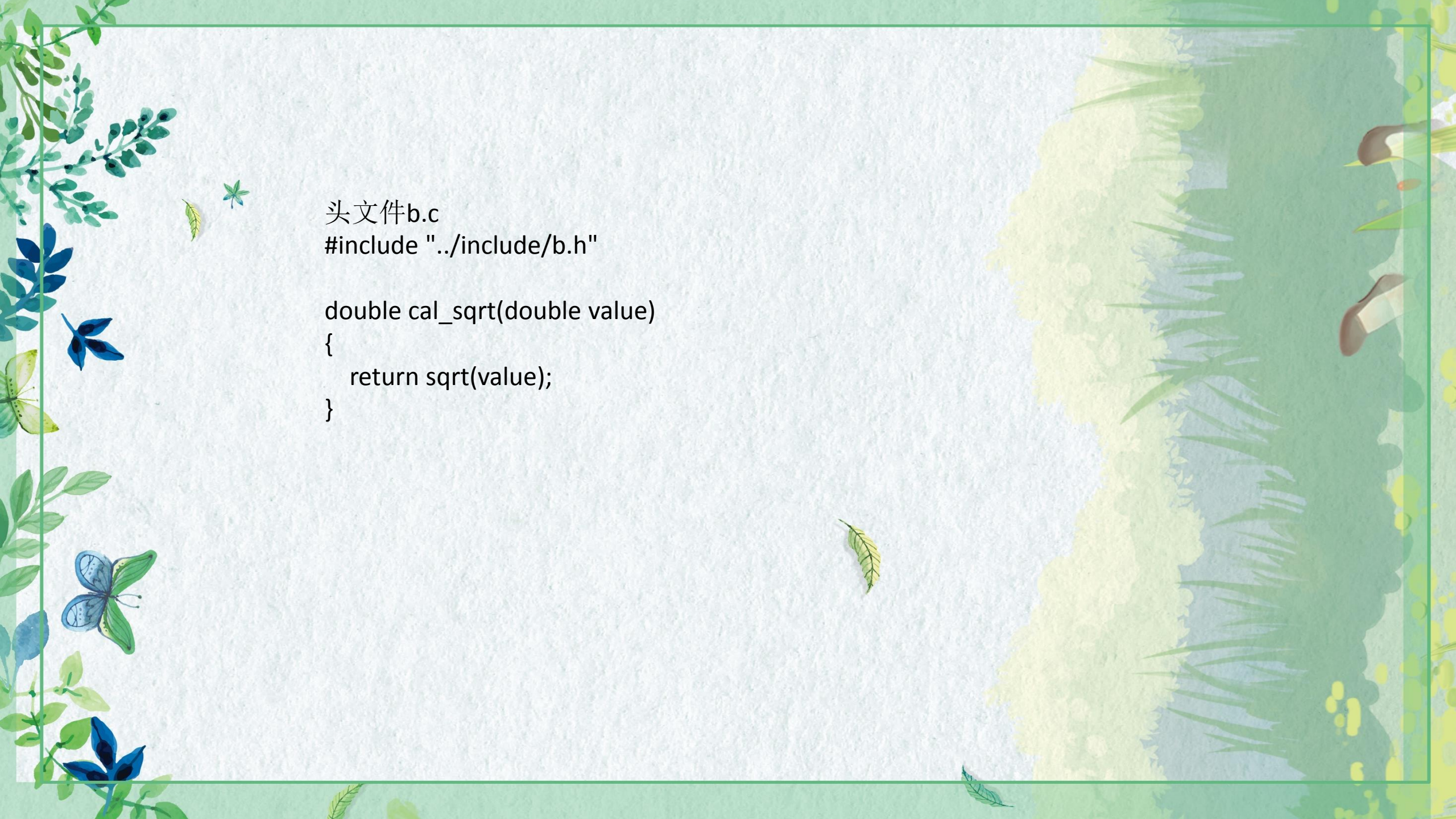
```
#define B_FIEL_HEADER_INC
```

```
#include<math.h>
```

```
double cal_sqrt(double value);
```

```
#endif
```





头文件b.c  
#include "../include/b.h"

```
double cal_sqrt(double value)
{
    return sqrt(value);
}
```



## 主函数main.c

```
#include "../include/b.h"  
#include <stdio.h>  
int main(int argc, char** argv)  
{  
    double a = 49.0;  
    double b = 0.0;  
  
    printf("input a:%f\n",a);  
    b = cal_sqrt(a);  
    printf("sqrt result:%f\n",b);  
    return 0;  
}
```



## 编写CMakeLists.txt

#1.cmake version, 指定cmake版本  
`cmake_minimum_required(VERSION 3.2)`

#2.project name, 指定项目的名称, 一般和项目的文件夹名称对应  
`PROJECT(test_sqrt)`

#3.head file path, 头文件目录  
`INCLUDE_DIRECTORIES(  
include  
)`

#4.source directory, 源文件目录  
`AUX_SOURCE_DIRECTORY(src DIR_SRCS)`



## 编写CMakeLists.txt

#5.set environment variable, 设置环境变量, 编译用到的源文件全部都要放到这里, 否则编译能够通过, 但是执行的时候会出现各种问题, 比如"symbol lookup error xxxxx , undefined symbol"

```
SET(TEST_MATH  
  ${DIR_SRCS}  
)
```

#6.add executable file, 添加要编译的可执行文件  
ADD\_EXECUTABLE(\${PROJECT\_NAME} \${TEST\_MATH})

#7.add link library, 添加可执行文件所需要的库, 比如我们用到了libm.so (命名规则: lib+name+.so), 就添加该库的名称  
TARGET\_LINK\_LIBRARIES(\${PROJECT\_NAME} m)



编译和运行程序：

准备好了以上的所有材料，接下来，就可以编译了，由于编译中出现许多中间的文件，因此最好新建一个独立的目录**build**，在该目录下进行编译，编译步骤如下所示：

```
mkdir build  
cd build  
cmake ..  
make
```



# 作业

---

完成嵌入式下采集单目摄像头的图像并测量出摄像头与视野中心附近二维码的距离

提示：

1. 二维码识别可以用ZBar这个库来实现
2. 距离测量的方法，三角测量、相似三角形测量
3. 需输出单目相机的内参
4. 可以想想如何把上次的程序在Ubuntu下运行



THANK YOU  
感谢各位观看