所需头文件

```
#include <fcntl.h>  //文件控制选项头文件
#include <stdio.h>
#include <termios.h>  //linux串口相关的头文件
#include <unistd.h>
#include <stdlib.h>
```

读取数据流程

1. 给串口设备赋权，使它可以读写

命令:

```
sudo -S chmod 777 <device-name>
```

我们每次接上设备的时候都要在终端记得运行这条命令。

也可以把命令写到代码里面。

```
system(const char *command)  // #include <stdlib.h>
```

system可以运行命令。

2. 配置设备

打开串口

```
int fd = open(const char *file, int oflag); // #include <fcntl.h.h>
```

第一个参数是设备名称，是 char*类型的，需要是绝对路径，如 "/dev/ttyUSB0"。

第二个参数是读取模式，有挺多的，可以去网上查一下，我们一般是

使用 O_RDWR | O_NOCTTY

- O_RDWR 是可读可写
- O_NOCTTY 按blog来说是，如果file指的是终端设备，则不将此设备分配作为此进程的控制终端。
- O_NONBLOCK 如果file指的是一个F I F O、一个块特殊文件或一个字符特殊文件，则此选择项为此文件的本次打开操作和后续的I / O操作设置非阻塞方式，**我测试过，没什么用** 。

open函数的返回值是int，是一个文件描述符。他就代表这个串口。

配置

```
struct termios options;
// 先获取fd的属性结构体
tcgetattr(fd, &options); // #include<termios.h>

options.c_iflag = IGNPAR;                                //输入模式
```

```
options.c_oflag = 0;                                    //输出模式
options.c_cflag = baud_rate | CS8 | CLOCAL | CREAD;  //控制模式
options.c_lflag = 0;                                    //本地模式
options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);

options.c_cc[VTIME] = 0;  //最少读取字符数
options.c_cc[VMIN]  = 1;  //超时时间, 单位: 100ms

tcsetattr(fd, TCSANOW, &options);  //应用上面的设置
```

```
IGNPAR: 忽略奇偶校检错误的字符。
baud_rate: 比如波特率是115200, 就是B115200
CS8: 8 位数据位
CLOCAL:  忽略调制调解器状态行
CREAD:  启动接收
```

主要配置是波特率，其他的感觉没什么需要改的。

3. 读取与发送

```
int n = read(int fd, void *buf, size_t nbytes);
int n = write(int fd, const void *buf, size_t n)
```

读取和发送都是三个参数，文件描述符、数组、字节长度。

因为都是以字节为单位的，我们使用 unsigned char、 uint8_t 数组。

这两者是等价的。

读取

```
unsigned char buff[10];
int n = read(fd, buff, 2);
```

buff是指针，read读取到数据就会存到buff，然后buff++，接着往下存。

read的返回值是读取数据的字节数。

有时读取的数据小于n时，没数据读了，read函数也会结束，就会出现返回值小于n的情况。

write 没什么说的，把要发送的数据存到数组，指定长度就可以了。

任务

发送数据格式为：

长度4字节：帧头 + **x** +帧尾，x为 uint16_t 类型，递增发送。

0: 帧头 = 0xFF

1: x的高八位

2: x的低八位

3: 帧尾 = 0xFE

读取流程

1. 读取帧头
2. 判断帧头是否正确
3. 读取剩余内容
4. 判断帧尾是否正确
5. 保存数据

```
while:
    if read(fd, buff, 1) == 1:
        if buff[0] == 0xFF:
            if read(fd, buff + 1, 3) == 3:
                if buff[3] == 0xFE:
                    x...
```

数据转化

x是 uint16_t 类型的，如何使用两个 uint8_t 的数据转化成 uint16_t 的数据，方法:

1. 使用union

```
union a{
    uint8_t i[2];
    uint16_t o;
};
```

union只占一份内存。

1. 使用 移位 <<

先类型转化, << 8位, 加上低八位。

要求

1. CMake
2. 使用两种代码架构

> src/
>
> serial.cpp
>
> include/
>
> serial.hpp
>
> main.cpp
>
> CMakeLists.txt

第二个需要使用子cmake，add_subdirectory()

> serial/
>
> serial.cpp
>
> serial.hpp
>
> CMakeLists.txt
>
> main.cpp
>
> CMakeLists.txt

3. main

在main函数里面创建一个死循环，使用opencv随便创建一个Mat，把串口读取的数据画到Mat上，休眠20ms（模拟图像处理耗时），imshow出来

休眠可以使用 C++ thread库，或者 sleep()。