实验二　文件 IO 和串口通信

一、实验目的：

1. 了解 IO 缓存的基本概念
2. 熟练掌握打开、创建、读、写，关闭等操作文件的方法
3. 了解串口基本概念
4. 熟练掌握串口的初始化、打开、发送、接收等

二、实验器材：

iTop-4412 实验开发板一套、配套交叉串口线一根、PC 机一台（Windows 操作系统）、U 盘一个（FAT 格式）

三、实验原理：

1.文件 IO

“Linux 中一切皆文件”。在 linux 系统中，所有的 I/O 操作都是通过读文件或者写文件来完成的。在这里，把所有的外围设备，包括串口、蜂鸣器、led、键盘和显示器等等，都看成是文件系统中的文件。文件操作主要涉及到打开、关闭、读、写四个函数。

a) 文件打开函数——open 函数

open 函数可以建立一个到文件或者设备的访问路径。在打开或创建文件时可以制定文件的属性及用户的权限等各种参数。使用 open 函数的时候会返回一个文件句柄，文件句柄是文件的唯一识别符 ID。对文件的操作必须从读取句柄开始。返回句柄。open 函数有两个原型，

```
int open(const char *path, int oflags);
int open(const char *path, int oflags,mode_t mode);
```

第一个参数 path 表示：路径名或者文件名。路径名为绝对路径名，例如开发板中的串口 3（靠近耳机和麦克风的串口）驱动的设备结点/dev/ttySAC3。

第二个参数 oflags 表示：打开文件所采取的动作。

必选参数	O_RDONLY	文件只读
	O_WRONLY	文件只写
	O_RDWR	文件可读可写
可选参数	O_APPEND	每次写操作都写入文件的末尾
	O_CREAT	如果指定文件不存在，则创建这个文件
	O_EXCL	如果要创建的文件已存在，则返回 -1，并且修改 errno 的值
	O_TRUNC	如果文件存在，并且以只写/读写方式打开，则清空文件全部内容
	O_NOCTTY	如果路径名指向终端设备，不要把这个设备用作控制终端。
	O_NONBLOCK	如果路径名指向 FIFO/块文件/字符文件，则把文件的打开和后继 I/O 设置为非阻塞模式（nonblocking mode）
	O_NDELAY	和 O_NONBLOCK 功能类似，调用 O_NDELAY 和使用的 O_NONBLOCK 功能是一样的。

第三个参数 mode 表示：设置创建文件的权限。

S_IRUSR,S_IWUSER,S_IXUSR,S_IRGRP,S_IWGRP,S_IXGRP,S_IROTH,S_IWOTH,S_IXOTH.

其中 R：读，W：写，X：执行，USR：文件所属的用户，GRP：文件所属的组，OTH：

其他用户。第三个参数可以直接使用参数代替,0777 表示所有用户的所有权限。

b) 文件关闭函数——close 函数

调用 close 函数之后，会取消 open 函数建立的映射关系，句柄将不再有效，占用的空间将被系统释放。函数原形为 int close(int fd);

参数 fd，使用 open 函数打开文件之后返回的句柄。

返回值，一般很少使用 close 的返回值

c) 文件读函数——read 函数

read 函数在头文件“#include <unistd.h>”中。函数原型为

ssize_t read(int fd，void *buf,size_t len)

参数 fd，使用 open 函数打开文件之后返回的句柄。

参数*buf，读出的数据保存的位置。

参数 len，每次最多读 len 个字节。

返回值为 ssize 类型，出错会返回-1，其它数值表示实际写入的字节数，返回值大于 0 小于 len 的数值都是正常的。

d) 文件写函数——write 函数

write 函数在头文件“#include <unistd.h>”中。函数原型为

ssize_t write(int fd，const void *buf,size_t count)

参数 fd，使用 open 函数打开文件之后返回的句柄。

参数*buf，需要写入的数据。

参数 count，将参数*buf 中最多 count 个字节写入文件中。

返回值为 ssize 类型，出错会返回-1，其它数值表示实际写入的字节数。

2.串口通信

串口通信是指一次只传送一个数据位。虽然在通信的时候串口有 8 位或者 9 位等，但是在物理层面传输的时候，它仍然是以单个 bit 的方式传输的。串口也是文件，所以串口的通信和文件 IO 一样，也是打开、关闭、读、写操作。串口的操作流程如下图所示，

打开串口，一般使用 open 函数，打开之后会返回句柄，这个句柄就可以提供给发送和接收函数使用。串口本质上也是字符设备，但是串口是属于一种比较特殊的字符设备。

初始化串口，串口需要配置波特率，数据位，校验位等等一系列的参数，初始化的过程掌握了，发送和接收都比较容易实现。初始化比较麻烦，本次实验中已经给出了封装好的初始化函数 int set_opt(int fd_uart,int nSpeed, int nBits, char nEvent, int nStop)。只要能够读懂代码，根据实际需求进行验证和配置即可。

对串口进行初始化，主要是对以下结构体进行赋值。

```
#define NCC 8
struct termio {
    unsigned short c_iflag;     /* input mode flags */
    unsigned short c_oflag;     /* output mode flags */
    unsigned short c_cflag;     /* control mode flags */
    unsigned short c_lflag;     /* local mode flags */
    unsigned char c_line;       /* line discipline */
    unsigned char c_cc[NCC];    /* control characters */
};
```

初始化流程如下，

函数 tcgetattr 用于读取当前串口的参数值，在实际应用中，一般用于先确认该串口是否能够配置，做检测用。使用这个函数前可以先定义一个 termios 结构体，用于存储旧的参数。函数原型为 int tcgetattr(int fd, struct termios *termios_p)。

参数 1：fd 是 open 返回的文件句柄。

参数 2：*termios_p 是前面介绍的结构体。

函数 cfsetispeed 和 cfsetospeed 用于设置波特率。执行成功返回 0，失败返回-1。函数原型 int cfsetispeed(struct termios *termios_p, speed_t speed);

参数 1：*termios_p 是前面介绍的结构体。

参数 2：speed 波特率，常用的 B2400，B4800，B9600，B115200，B460800 等等

函数 tcflush 用于清空串口中没有完成的输入或者输出数据。在接收或者发送数据的时候，串口寄存器会缓存数据，这个函数用于清除这些数据。执行成功返回 0，失败返回-1。原型为 int tcflush(int fd, int queue_selector);

参数 1：fd 是 open 返回的文件句柄。

参数 2：控制 tcflush 的操作。有三个常用数值，TCIFLUSH 清除正收到的数据，且不会读取出来；TCOFLUSH 清除正写入的数据，且不会发送至终端；TCIOFLUSH 清除所有正在发生的 I/O 数据。

tcsetattr 函数是设置参数的函数。执行成功返回 0，失败返回-1。原型为 int tcsetattr(int fd, int optional_actions,const struct termios *termios_p);

参数 1：fd 是 open 返回的文件句柄。

参数 2：optional_actions 是参数生效的时间。有三个常用的值：TCSANOW：不等数据传输完毕就立即改变属性；TCSADRAIN：等待所有数据传输结束才改变属性；TCSAFLUSH：

清空输入输出缓冲区才改变属性。

参数 3：*termios_p 在旧的参数基础上修改的后的参数。一般在初始化最后会使用这个函数。

发送和接收数据，前面提到过串口是属于字符设备的，可以使用 read 函数和 write 函数实现。

关闭，使用函数 close 即可关闭串口。

3.常用字符串函数

int sprintf(char *str, const char *format, ...)，函数：第一个参数为输出字符串的缓冲区，第二个第三个参数是字符串格式及组成素材，类似于 printf 函数的格式参数，例如"%s abc %d"这个格式就表明希望打印出一个字符串+abc+一个整型组合的字符串。sprintf 与之很类似，只不过最终生成的字符串会写入第一个参数中。

四、实验内容

串口监视器：将实验一中采集到的电阻值以及报警情况（正常、过低、过高）通过 UART1 以如下协议发送到电脑的串口调试助手上以 **16 进制显示（HEX）**。每发送一次阻值帧，都要附带一帧报警信息。注意 CRC 校验为提高部分，可借用网上现成的函数。后续不会给出 CRC 校验具体的值，因为 CRC 校验计算方式较多，只需保证收到帧的 CRC 与采用的 CRC 校验码计算方式计算出来结果相同。

数据类型（8 位整数，0 表示阻值，1 表示报警）	数据长度（后文数据区的长度，不包括 CRC）	数据区（长度根据数据长度而定）	CRC 校验（**提高**，校验除了 CRC 之外的整帧数据）
1 字节	1 字节	数据长度个字节	2 字节

其中，数据区做如下约定：对于阻值，直接传输一个 8 位整数，表示多少百Ω，比如 8.2kΩ就传输 82。对于报警信息，则约定正常传输“OK”，过低传输“LO”，过高传输“HI”。

那么如果传输一帧阻值为 8.2kΩ的信息，信息流为（全为 16 进制表示，两个十六进制表示一个字节，0x52 就是 82 的 16 进制表示）：

0x00 01 52 【对 0x00 01 52 进行运算后得到的两字节 CRC】

如果传输一帧高阻值的报警信息，信息流为（其中 4C 4F 为 L 和 O 的 ASCII 码）：

0x01 02 4C 4F 【对 0x01 02 4C 4F 进行运算后得到的两字节 CRC】

为了避免串口调试助手内帧粘连过于严重，建议降低发送频率，如 1s 一次。

文件 IO：打开文件，将第一问中原本通过串口发送的内容改为在文件中写入。写入时，需要保证有至少 5 次阻值数据。然后读取该文件的内容，并打印出解码后的数据。即只打印出与下面三个字符串类似的字符串，注意换行：Resistance：8200，Alert：OK； Resistance：9200，Alert：HI；Resistance： 200，Alert：LO。

提高：按前两个实验的说明，加入 CRC 校验部分。因为 CRC 校验计算方式较多，只需保证收到帧的 CRC 与采用的 CRC 校验码计算方式计算出来结果相同，也即计算后，采用现成的工具进行验算即可。

五、实验步骤

1. 完成串口准备实验：看懂串口初始化函数，修改 uart.c，以 115200，8N1 配置打开串口，并可以**不停地**发送字符串“hello”。编译完成后，用 U 盘拷入开发板并且运行。然后将板子上的串口替换到标有 UART1 的串口上，观察串口输出；
2. 完成串口通信实验：整合实验一的代码和串口源码 uart.c，完成串口监视器的功能。编

译完成后，用 U 盘拷入开发板并且运行。然后将板子上的串口替换到 UART1，观察串口输出；

3. 文件 IO 准备实验：根据 fileIO.c，尝试对文件进行先写入后读取，编译完成后，用 U 盘拷入开发板。将串口线连接到实验板串口，另一端连接 PC 机，使用超级终端运行程序；
4. 完成文件 IO 实验：根据 fileIO.c，结合实验一的代码，完成文件 IO 中要求的功能，编译完成后，用 U 盘拷入开发板。将串口线连接到实验板串口，另一端连接 PC 机，使用超级终端运行程序；
5. 完成提高部分，加入 CRC 校验内容。

六、实验注意事项：

最好不要热拔插串口，如果强行热拔插，容易损坏串口芯片。

要注意串口收发双方的波特率和奇偶校验、停止位、数据位等设置都必须相同。