项目执行相关

使用了Cmake，方便生成Makefile对项目进行构建，这里有一片详细教程 CMake入门实战，大致能看懂CMakeLists.txt就行，不看也没关系，会编译执行程序就好，下面是程序编译执行的过程

实验中，server和client共用库函数，所以就在最外面的CMakeLists里面生成了两个可执行文件

```
1. 进入build文件夹下，执行
cmake ../
会在build文件夹下生成项目的Makefile文件
2. 然后执行
make
编译可执行程序，该命令会在build下生成可执行程序server和client
3. 运行程序，在build下执行（两个终端下，分别执行）
./server // 使server.cc进程启动
./clinet // 使client.cc进程启动
```

实现思路

这里列出的顺序，就是写代码的顺序，你可以照着这个顺序去理解代码

不能特别详细的解释每个细节，但思路都捋了一遍，其中的具体代码都是很简单的代码，很容易看懂

一、如何模拟两端通信

选择了`unix socket IPC`来实现进程间的通信，用法和socket网络编程一样（同一套接口，只是使用范围不同）`socket_IPC.cc`文件解释：

- 建立两进程间通信的函数在`socket_IPC.cc`内，调用在client.c和server.c中
- pthread_recv和pthread_send函数（`socket_IPC.cc`里），是socket通信的两个函数，并不代表HDLC通信，HDLC通信会依据这两个函数进行模拟

> HDLC并不区分客户端和服务端（采用了异步平衡方式），名字只是为了区分两个进程，两个主函数公用了src里面编译生成的动态库（也就是说`server.c`和`client.c`是相互独立的，都有main函数）

二、实现HDLC连接建立的功能

`askHDLC`函数，请求建立HDLC连接 先发送请求，然后监听信道，看对方是否发回确认帧

`reAskHDLC`函数，等待对方发起建立连接的请求 刚开始是一个死循环，监听对方的请求帧；收到请求后，进行各种合法性验证，然后发送确认帧。

这里还用到了U_Framing函数，用于无编号帧的构造，无非就是根据帧结构，设置各字段

> 这里和实际的异步平衡HDLC不太一样....程序中没办法两端都直接响应对方的请求帧，所以只能让两程序调用不同的函数

三、实现HDLC的查链过程

这个部分也可以放在后面实现，不过和建立连接过程基本一样，所以就一起写了 很容易理解，就是传参的值改变了一下

四、HDLC通信 帧发送 的实现

1. 由于考虑到对帧Flag的使用，所以，我通过对要发送消息msg的划分（设置了最大帧长，但Info部分仍为可变，只是限制了不让其无限大，但这也代来了缺点，就是前面好几帧长度都一样，只有最后一帧可能更短些）

> 这个功能在splitInfo函数里实现，对要发送的msg划分，调用I_Framing组帧，组帧成功后，加入到发送队列sendQueue

2. 然后对划分的msg进行组帧，调用的是I_Framing函数 很简单的一个函数，还是对帧的各字段进行赋值
3. I_Framing里面调用了setControl函数 该函数中，通过全局变量sWin对control字段进行赋值；之后，还在该函数内进行了窗口的更新

> sWin代表发送窗口，rWin代表接收窗口，他们同属于同一个进程 **有一点不要搞混，server和clinet属于两个不同进程，他们不会共用全局变量！！！**

> **一些变量的解释** buffer: 这个是模拟信道中的信息，帧在信道中是相互挨着，所以接收方才需要根据Flag进行划分，buffer就是起到模拟信道数据的功能 msg: 用户要发送的信息（在终端输入的信息） frame: 对划分的msg封装得到的帧

五、HDLC通信 帧接收 的实现

1. 从pthread_recv函数中收到另一进程发来的信息buffer（信道中得到的信息）
2. 调用divideBuf，根据FLag对buffer划分，得到一个个帧frame
3. 验证帧的合法性（是否发给我，FCS是否正确等....）

> 理解思路时，可以先不管监督帧相关的（比如：FCS校验不通过后，发送srej帧），这个我是在最后实现的,可以跟着写代码的思路来理解代码

> divideBuf最后有个if，这个是发送rr帧的，也是放在后面再看

六、监督帧发送相关功能的实现

对各种错误的处理，监督帧的构造发送都是最后实现的 这里并没有实现所有的监督帧`RR，RNR，REJ，SREJ`，这里实现了`RR，SERJ`

- **RR帧**，对按序收到的帧，发送确认帧，调用位置在divideBuf最后那个if
- **SREJ帧**，在FCS校验不通过时调用S_Framing函数，发送srej帧

> S_Framing会根据传参，来构造发送不同的帧

七、接收到监督帧应采取的处理

1. 对监督帧进行响应，就要先得到监督帧，所以`deal_SFrame`的调用在divideBuf中（帧合法性检验处），对control进行判断，看是否为监督帧，若是，则调用`deal_SFrame`
2. `deal_SFrame`函数，主要用来判断S_Framing的类型，然后调用不同处理函数

处理函数有两个

1. feedback_RR，对RR帧处理 RR帧其实就是起到更新窗口的作用，还有对发送缓存sendCache的清理
2. feedback_SREJ 在sendCache中，找到要重发的帧，并将其加入到发送队列中 这里因为采用的数据结构为队列，所以出队的数据都找不回来（感觉影响不大，毕竟该程序中收到srej3，说明3之前都收到了）

其他

- FCS校验和 在计算校验和时，代码中可能有两种方式，emmm，有一种是错的。书上校验和的计算：从control开始，到fcs开始前的部分参与计算 但写的过程中，有时候习惯直接将【帧开始，到fcs前】这部分进行计算了.... 虽然计算方法不对，但计算也接收时校验都是对应的，所以并不会导致FCS校验不通过（看兴趣吧，想改的话，自己修改一下）

```
fcs = countFcs((unsigned char *)buf_pter, (unsigned int)(buffer -
buf_pter)); // 不对的
fcs = countFcs((unsigned char *)(buf_pter+1), (unsigned int)(buffer -
buf_pter - 1)); // 正确的
```

- 输出问题 模拟过程的输出最好改一下吧，两组实验的输出是完全一样的（而且这里面有一些调式时的输出2333）
- 命名问题 由于起名字很痛苦，而且容易先入为主，所以这个项目和我们的项目函数名差不多，而且风格有点乱（有使用下划线的，还有使用驼峰的） 有些函数名字表达的意思也不太准确（呃，时间总不能花在想名字上） 所以，如果你觉得有更合适得名字，或者想统一下命名风格，可以把这些优化一下
- 最最后：基本功能实现了，展示看起来没问题。但还有许多可以优化的地方：
    1. 窗口并不对发送帧的数量进行限制（仅仅是在接收端判断窗口不够，直接抛弃），信号量的相关函数在signal_lib.c/h里面，没用233333，如果你不想改，直接把两个文件删除就行
    2. 虽然采用了srej，但是最后并没有对收到的帧进行排序（简单的说就是，我发送了123456789，如果出现了重传等情况，模拟的接收端并不能恢复123456789）不过这点老师看不出来，因为打印数据是一帧一帧打印的，并没有打印整条msg

可供参考的实验报告

报告原理，函数等的顺序可按照这个文件叙述过程来写

目录

一、协议介绍……

二、实验内容……

三、实现思路……

四、实验环境……

五、代码框架……

六、实现细节

1. socket 通信 ……
2. 组帧 ……
3. 建立链接，拆除链接 ……
4. 流量控制：滑动窗口 ……
5. 帧校验（CRC 冗余校验）……
6. 差错控制：选择拒绝 ARQ……
7. 差错模拟……

七、结果展示

1. 执行步骤……
2. 正常执行……
3. 错误模拟……

八、实验中遇到的问题……

九、实验感悟……