整体思路

该项目主要分为客户端和服务器两大模块，客户端只用来处理客户的操作，并将数据发送给服务器，并且客户端不能直接访问存储的数据，所有数据由服务器访问并修改，并通过消息队列发送给客户端。服务器通过创建子程序，每个子程序调用不同的可执行文件来实现银行终端的功能。

模块分析

common.h头文件

包含了整个项目所需的头文件和数据类型

a. C2SKEY和S2CKEY：客户端和服务器端消息队列创建时的键值

b. enum opt {OPEN = 1, DEP, DISDEP, CHECK, TRANS, DELAC, QUIT};//开户 存款 取款 查询 转账 销户 退出，定义了枚举菜单，消息队列通信时的消息类型

c. 定义了传递消息的数据结构:

1. DATA:账号信息的结构体 账号 户主 余额 密码 转账账户
2. C2SMSG:客户端到服务器的消息格式 data结构体和 tyoe消息类型
3. S2CMSG:服务器到客户端的响应消息格式 type消息类型， 账号 余额 户主

serve.c 服务器

主进程通过 for 循环来循环调用 fork 函数创建子进程，每个子进程独立处理一项功能，每个子进程通过消息队列和客户端进行通信，并使用消息改造函数改造二号信号，当服务器收到二号信号时，将会杀死所有子程序，并且删除消息队列，退出服务器

main()函数

i. 创建消息队列

ii. 根据 infor 数组，通过fork函数创建子进程，infor 数组包含可执行文件的路径和文件名

iii. 每个子进程调用execl函数来执行相应的可执行文件，实现相应功能

iv. 主进程等待二号信号来调用stop()函数

stop()函数

i. 收到二号信号时被调用

ii. 遍历杀死每个子进程

iii. 删除消息队列

iv. 退出服务器

client.c 客户端

创建消息队列和服务器进行通信，通过菜单界面提示用户相应的功能并调用相应的功能函数，功能函数会根据用户的输入发送消息给服务器来获取相应的数据，但客户端本身并不能操作数据,客户端接收服务器的消息结果并显示给用户，通过主循环来实现客户端的循环运行。

main()函数 循环调用，整个客户端框架

i. 访问服务器消息队列

ii. 循环调用 menu() 函数来显示菜单并获取选择

iii. 根据选择调用不同的功能函数

menu()函数 菜单

i. 通过枚举变量打印菜单

ii. 获取用户选择并返回相应编号

iii. 调用 glob() 函数来查询指定目录下的所有账号文件，每次菜单被调用的时候就刷新一次

CREATACCOUNT() 函数 开户

i. 获取用户输入的账号信息并判断是否合法

ii. 构造访问服务器的消息结构体

iii. 通过消息队列将创建的账号信息发送到服务器

iv. 接收并处理服务器返回的结果

v. 完成开户功能

dep() 函数 存款 disdep() 函数 取款 check() 函数 查询 delac() 函数 销户 trans() 函数 转账

这些函数思路基本一致，所以整体列出

i. 获取用户输入的账号

ii. 使用字符串拼接函数拼接成账号文件名，通过遍历来与 glob() 函数获取的文件名比较

iii. 查询到账号文件向服务器发送请求，服务器返回账号信息，否则打印错误信息并退出功能函数

iv. 根据账号信息来确定用户操作是否合法，如果非法则退出功能函数，并向服务器发送错误信息，使服务器的相应功能重置为初始状态方便下次访问，如果用户操作合法，向服务器发送操作后的账号信息，服务器接收并作出相应处理

v. 接收服务器的返回结果并打印

clear() 和 clears() 清理函数

i. 清理输入缓冲区和屏幕

open.c 服务器开户功能

通过消息队列接收客户端的账号信息，通过字符串拼接将银行卡号和 .txt 拼接起来作为账户i信息文件名，并创建相应的账号文件来保存信息

main() 函数

i. 创建消息队列

ii. 循环接收客户端的开户请求

iii. 调用 CREATID() 生成新账号

iv. 调用 save_info() 来保存文件

v. 构造响应消息,发送给客户端

CREATID() 函数 生成银行卡号

i. 打开账号ID记录文件id.txt

ii. 读取当前最大账号ID

iii. 递增生成新账号

iv. 写入新ID,返回给主函数

save_info()函数

i. 根据账号创建对应数据文件

ii. 将开户信息写入该文件

dep.c 服务器存款功能 disdep.c 服务器取款功能 check.c 服务器查询功能 trans.c 服务器转账功能 delac.c 服务器销户功能

这些模块功能基本一致，所以整体列出

a. 创建消息队列

b. 接收客户端所发送的账户文件名

c. 向客户端发送账户的具体信息，如银行卡号，余额，户主等等

d. 再次接收客户端所发送的信息，并进行判断，让如果是错误信息则初始化该模块功能，方便下次调用，如果是正确信息则继续服务器该功能，继续向下执行，最后返回处理结果给客户端

项目过程所遇到的bug

忘记考虑 == 运算符的优先级高于 =

```
if((fd = open("id.txt", O_RDWR | O_CREAT, 0664) == -1)){
                perror("1open err");
                return -1;
 }
if((read(fd, &init_id, sizeof(init_id))) == -1){
                perror("read err");
                return -1;
 }

 分析
 open("id.txt", O_RDWR | O_CREAT, 0664) == -1 的值为0，导致fd的值为0，而0表示标准输入，所以read从键盘读取内容，导致刚开始开户的时候升序卡住，等待标准输入，并且会导致刚开始的银行卡号不正确
```

客户端和服务器的一个功能里面进行了多次消息队列的读取和发送，客户端第一次读取服务器发送的消息，之后客户端在判断用户操做非法后，就退出了客户端的功能函数，但服务器并不知情，仍然在等待客户端功能函数第二次发送消息，会导致客户端和服务器的信息传递错位，所以在服务器第二次接收消息后要进行判断，如果为错误消息则重置该功能，否则继续运行

```
//用户操作非法客户端功能函数退出，并向服务器发送错误信息
//让id = 0 并且发送给服务器
if(num == 3){
```

```
                                clears();
                                printf("密码错误3次，已退出\n");
                                printf("按任意键继续\n");
                        printf("客户端存款功能准备发送错误信息\n");
                        r_buf.type = DISDEP;
                        r_buf.data.id = -1;
                        if(msgsnd(c2smsgid, &r_buf, sizeof(struct C2SMSG) - sizeof(long), 0) == -1){
                                                perror("msgsnd err");
                                                return;
                        }
    }

//接收消息队列信息
//并且判断id是否为 -1， 如果id == -1， 重置服务器该模块功能，方便下次调用
 printf("存款系统第二次接收信息开始\n");
 if(msgrcv(c2smsgid, &r_buf, sizeof(struct C2SMSG) - sizeof(long), DEP, 0) == -1){
                perror("msgrcv err");
                return 0;
 }
 printf("存款系统第二次接收信息完毕\n");
 if(r_buf.data.id == -1){
                continue;
 }
```

使用glob函数时，忘记剪切出文件的文件名， glob函数所获取的文件名，都带有地址，所以可以通过指针偏移来去掉路径

```
glob_t file = {0};
glob("/home/yyy/linux/2银行/src/*.txt", 0, NULL, &file); //获取所有账号文件
len = strlen("/home/yyy/linux/2银行/src/");
for(int i = 0; i < file.gl_pathc, i++){
    strcpy(file.gl_pathv[i], file.gl_pathv[i] + len);
}
```

SS