|  |
| --- |
| **07111206** |
| 短信收发系统设计文档 |

|  |
| --- |
| 组名：一又二分之一  组长：吴一凡  组员：段 然 訾 娟  宋生安 李佳琦  王煜辰 郝鑫宇 |

目录

[一、数据库设计 2](#_Toc401162105)

[1.数据结构 2](#_Toc401162106)

[2.程序设计 3](#_Toc401162107)

[（1） 服务器端编程 3](#_Toc401162108)

[（2） 客户端编程 4](#_Toc401162109)

[二、网络通信的接口设计 6](#_Toc401162110)

[1. 通信协议 6](#_Toc401162111)

[2. 通信方法 6](#_Toc401162112)

[3. 数据包结构 6](#_Toc401162113)

[a. 客户端->服务器 6](#_Toc401162114)

[b. 服务器->客户端 6](#_Toc401162115)

[4. 客户端接口 6](#_Toc401162116)

[5. 服务器接口 6](#_Toc401162117)

[（1） 发送接口 6](#_Toc401162118)

[（2） 接收接口 7](#_Toc401162119)

## 一、**数据库设计**

**说明**：

我们组的短信模拟系统服务器和客户端都是用数据库存数据，整个项目开始设计时是计划服务器端用MySQL实现，客户端用轻量级数据库SQLite实现。接下来的编程过程中，由于我们用的是C++，开发环境是VC++ 6.0 和 Microsoft Visual Studio 2013,配置连接MySQL一直没有成功，于是服务器端放弃使用MySQL改为SQL Server,

客户端用SQLite

**详情**：

### **1.数据结构**

(1)服务器数据库表：

1)用户信息表：

|  |  |  |
| --- | --- | --- |
| User\_key(用户号码) | User\_state(用户状态) | Money\_left(话费余额) |
| … | … | … |

2)短信缓存表

|  |  |  |  |
| --- | --- | --- | --- |
| User\_key\_sent(发信人号码) | User\_key\_receive(收信人号码) | Time(发信时间) | Message\_content(短信内容) |
| … | … | … | … |

（2）客户端数据库表：

1)通讯录表：

|  |  |
| --- | --- |
| Friend\_key(联系人号码) | Friend\_name(联系人姓名) |
| … | … |

2)短信内容存储表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User\_key(用户号码) | Friend\_key(好友号码) | M\_time(短信时间) | Message\_content(短信内容) | M\_flag(短信标志<是收件或发件>) |
| … | … | … | … | … |

### 2.**程序设计**

1. 服务器端编程：
2. API:ADO
3. 接口程序

|  |  |  |
| --- | --- | --- |
| 函数 | 传入参数 | 说明 |
| void on\_change\_state(char\* key) | 用户号码 | 用户上线，改变服务器数据库中用户信息表的用户状态<1变为0> |
| int check\_m(char\* key) | 用户号码 | 检查有多少条短信待接收并返回条数，主要用于刚上线时的检查 |
| void get\_message(char\* key) | 用户号码 | 从数据库中提取出待收信息并放到socket主程序的全局变量中 |
| void off\_change\_state(char\* key) | 用户号码 | 用户下线，改变服务器数据库中用户信息表的用户状态<0变为1> |
| char get\_state(char\* key) | 用户号码 | 读取该用户状态 |
| void message\_cache(char\* ks, char\* kr, char\* tm, char\* m) | 发信人号码，收信人号码，发信时间，短信内容 | 当收信人不在线是将传入的内容存到短信缓存表中 |

1. 客户端编程:
2. API: sqlite3
3. 接口程序：

|  |  |  |
| --- | --- | --- |
| 函数 | 参数 | 说明 |
| void opendb() | 无 | 创建/打开数据库文件 |
| void name2ph\_num(char\* n) | 联系人姓名 | 获取该联系人的号码放到客户端主程序全局变量中 |
| void select\_m(char\* uk,char\* fk) | 本人号码，好友号码 | 查找本号与该好友号码的所有短信记录 |
| void delete\_m1(char\* uk,char\* fk) | 本人号码，好友号码 | 删除本号与该好友的所有短信记录 |
| void delete\_m2(char\* uk,char\* fk,char\* mc)  { | 本人号码，好友号码，短信内容 | 删除本号与该好友的某条短信 |
| void receive\_insert(char\* uk,char\* ks,char\* mc,char\* tm) | 本人号码，好友号码，短信内容,短信时间 | 将收到的该好友的信息存入本号的数据库文件中，并标明为收件 |
| void sent\_insert(char\* uk,char\* kr,char\* mc,char\* tm) | void sent\_insert(char\* uk,char\* kr,char\* mc,char\* tm) | 将发给该好友的信息存入本号的数据库文件中，并标明为发件 |

## 二、**网络通信的接口设计**

1. 通信协议：TCP
2. 通信方法：SOCKET套接口
3. 数据包结构：
4. 客户端->服务器：信息类型（short型，长度2）+收信人号码/本机号码（char型，长度32）+信息内容(char型，长度不定)。
5. 服务器->客户端：信息类型（short型，长度2）+内容（char型，长度不定）。
6. 客户端接口:

（1）发送接口：

send(servSock, sendBuff, sizeof(sendBuff)+1, 0);

servSock是客户端建立的SOCKET接口；

sendBuff是客户端发送数据的缓存区；

（2）接收接口：

recv(cliSocket, recvBuf, 1024, 0);

cliSocket是客户端建立的SOCKET套接口；

recvBuf是客户端接收信息的缓存区，为char型；

1024为recvBuf的最大长度；

1. 服务器接口：
2. 发送接口：
3. 通信信息发送：

send(toSock, message, strlen(message) + 1, 0);

toSock是对应客户端所建立的SOCKET，即每收到一个socket连接请求，就立一个user结构，存储客户端信息，其中即包含用户对应的socket；

message即A发送给B的信息；

1. 登陆反馈信息发送：

send(cliSocket, loginInfo, strlen(loginInfo), 0);

cliSocket为服务器的SOCKET，也是登录用户的SOCKET，因为该函数用于用户登录时，此时的cliSocket被赋给用户的SOCKET；

loginInfo为登录反馈信息；

1. 接收接口：

recv(cliSocket, recvBuf, 2000, 0);

cliSocket在这里为服务器SOCKET；

recvBuff为服务器接收信息的缓存区；