

[文档标题]

Linux C聊天室

**目录**

[**一、项目背景--------------------------------------------3**](#一、项目背景)

[**二、系统分析--------------------------------------------4**](#二、系统分析)

[**三、主要代码分析---------------------------------------7**](#三、主要代码分析)

**[四、系统测试--------------------------------------------17](#四、系统测试)**

[**五、结束语----------------------------------------------23**](#五、系统测试)

**[六、参考文献--------------------------------------------24](#六、参考文献)**

**[七、附录（主体代码）------------------------------------25](#七、附录（主体代码）)**

**一、项目背景**

聊天工具又称IM软件或者IM工具，主要提供基于互联网络的客户端进行实时语音、文字传输。从技术上讲，主要分为基于服务器的IM工具软件和基于P2P技术的IM工具软件。大部分的即时通讯服务提供了状态信息的特性──显示联络人名单，联络人是否在线及能否与联络人交谈。在网络无所不在的今天，在Internet上，有ICQ、MSN、Gtalk、OICQ等网络聊天软件，极大程度的方便了世界各地的友人们的相互联系，也使世界好像一下子缩小了，变成了一个世界村。不管身处何处，连接上网，便可通过客户端（C/S架构），或者浏览器（B/S架构）与他人沟通。

Linux，全称GNU/Linux，是一套免费使用和自由传播的类Unix操作系统，其内核由林纳斯·本纳第克特·托瓦兹于1991年第一次释出，它主要受到Minix和Unix思想的启发，是一个基于POSIX和Unix的多用户、多任务、支持多线程和多CPU的操作系统。它能运行主要的Unix工具软件、应用程序和网络协议。它支持32位和64位硬件。Linux继承了Unix以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。Linux有上百种不同的发行版，如基于社区开发的Debian、archlinux，和基于商业开发的RedHat Enterprise Linux、SUSE、Oracle Linux等。

正是由于Linux的广泛使用，开发一款基于Linux的简单实用聊天室系统对刚刚接触该系统的人来说十分有必要，极大的有利于用户间的沟通。

**二、系统分析**

项目需求：  
1、注册功能

2、登录功能

3、私聊

4、群聊

5、密码密文处理

附加功能（本人自加的功能）：

6、表情包

7、个性签名

8、查看聊天记录

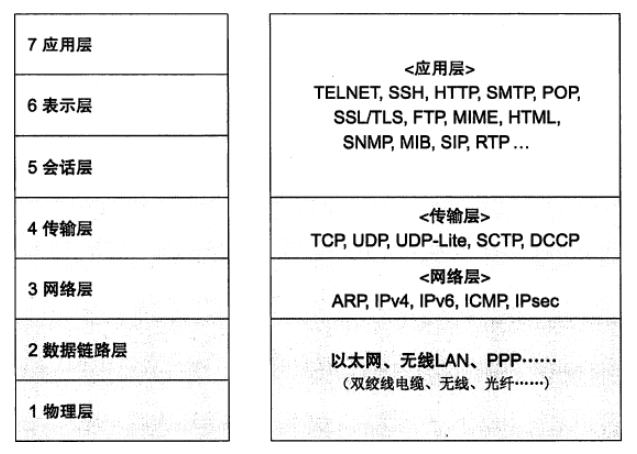
9、查看在线人数

10、修改个性签名

11、修改密码

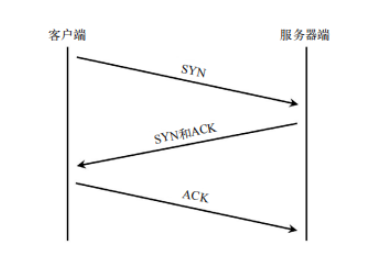
大致介绍一下项目主要用到的技术

OSI模型：



TCP三次握手过程：

客户端向服务器发起一个SYN包，服务器端返回对应的SYN的ACK响应以及新的SYN包，然后客户端返回对应的ACK。（在客户端和服务器之间建立正常的TCP网络连接时，客户端首先发出一个SYN消息，服务器使用SYN+ACK应答表示接收了这个消息，最后客户端再以ACK消息响应）。



SYN是同步序列编号，是TCP/IP建立连接时使用的握手信息。ACK是确认字符，在数据通信中，接收站发给发送站的一种传输类控制字符。表示发来的数据已确认接收无误。在TCP/IP协议中，如果接收方成功的接收到数据，那么会回复一个ACK数据。通过ACK信号有自己固定的格式，长度大小，由接收方回复给发送方。

第一次握手，建立连接时，客户端发送SYN包到服务器，并进入SYN\_SENT状态，等待服务器确认，其中SYN就是同步序列编号。第二次握手，服务器收到SYN包，必须确认客户的SYN，同时自己也发送一个SYN包，即是SYN+ACK包，此时服务器进入SYN\_RECV状态。第三次握手，客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK，此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。

TCP与UDP区别：

TCP(传输控制协议)提供的是面向连接、可靠的字节流服务。当客户端和服务器彼此交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。TCP提供超时重发，丢弃重复数据，检验数据，流量控制等功能，保证数据能从一端传到另一端。

UDP(用户数据报协议)是一个简单的面向数据报的运输层协议。UDP不提供可靠性，它只是把应用程序传给IP层的数据报发送出去，但是并不能保证它们能到达目的地。由于UDP在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快。

本项目两种方案都采用，TCP/IP体现在客户端和服务器的连接的建立，而UDP协议体现在群发功能的实现。

多线程（multithreading），是指从软件或者硬件上实现多个线程并发执行的技术。具有多线程能力的计算机因有硬件支持而能够在同一时间执行多于一个线程，进而提升整体处理性能。具有这种能力的系统包括对称多处理机、多核心处理器以及芯片级多处理或同时多线程处理器。在一个程序中，这些独立运行的程序片段叫作“程序”（Thread），利用它编程的概念就叫作“多线程处理”。

本项目为多个客户端同时使用，采用了多线程功能，实现读写分离，以及多客户端运行。

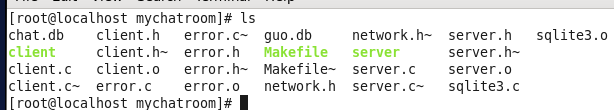
套接字：

所谓套接字(Socket)，就是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象。一个套接字就是网络上进程通信的一端，提供了应用层进程利用网络协议交换数据的机制。从所处的地位来讲，套接字上联应用进程，下联网络协议栈，是应用程序通过网络协议进行通信的接口，是应用程序与网络协议根进行交互的接口

Socket(套接字)可以看成是两个网络应用程序进行通信时，各自通信连接中的端点，这是一个逻辑上的概念。它是网络环境中进程间通信的API(应用程序编程接口)，也是可以被命名和寻址的通信端点，使用中的每一个套接字都有其类型和一个与之相连进程。通信时其中一个网络应用程序将要传输的一段信息写入它所在主机的 Socket中，该 Socket通过与网络接口卡(NIC)相连的传输介质将这段信息送到另外一台主机的 Socket中，使对方能够接收到这段信息。 Socket是由IP地址和端口结合的，提供向应用层进程传送数据包的机制

**三、主要代码分析**

代码目录目录(用户.db为系统为每个登录用户创建的数据库文件，chat.db为聊天记录,client server为用户端和服务器端)



为了实现S/C架构即服务器端和客户端，主要用到了以下函数

Socket函数：

#include <sys/types.h>

#include <sys/socket.h>

int socket(int domain, int type, int protocol);

/\*创建成功的返回值为整型的套接字描述符，创建失败则返回-1\*/

·domain：套接字的协议类型，socket函数支持的协议类型如下

协议名称 描述

AF\_UNIX, AF\_LOCAL 本地交互协议

AF\_INET IPv4协议

AF\_INET6 IPv6协议

AF\_NETLINK 内核接口设备协议

AF\_IPX IPX-Novell协议

AF\_X25 ITU-T X.25/ISO-8208协议

AF\_AX25

业余无线电AX.25协议

AF\_ATMPVC 原始ATM接入协议

AF\_APPLETALK 苹果公司的Appletalk协议

AF\_PACKET 底层数据包接口

·type：用于指定当前套接字类型，socket函数支持的套接字类型包括SOCK\_STREAM（数据流）、SOCK\_DGRAM（数据报）、SOCK\_SEQPACKET（顺序数据报）、SOCK\_RAW（原始套接字）、SOCK\_RDM（可靠传递消息）、SOCK\_PACKET（数据包）。

除了在使用原始套接字以外，通常情况下设置为0，以表示使用默认的协议。

在Linux系统中创建一个套接字时会在内核中创建一个套接字数据结构，然后返回一个套接字描述符标识这个套接字数据结构。这个套接字数据结构包含连接的各种信息，如目的地址、TCP状态以及发送接收缓冲区等。TCP协议这个套接字数据机构的内容来控制这条连接。

Bind函数：

在创建了套接字之后需要将本地地址和套接字绑定在一起，Linux提供了bind函数进行这个操作：

int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen); //若绑定成功则返回0，否则返回-1

其中参数sockfd是创建套接字时得到的套接字描述符，addr是本地地址，addrlen是套接字对应的地址结构长度。在C/S模式下服务器和客户端都可以使用bind函数来设置套接字地址，通常来说有以下5种模式：

·服务器指定套接字地址的公认端口号，不指定IP地址。这时addr的值设为INADDR\_ANY，表示它愿意接收来自任何网络设备接口的客户端连接，这是服务器最经常使用的绑定方式。

·服务器指定套接字地址的公认端口号和IP地址，表示服务器只接收来自对应于这个IP地址的特定网络设备端口的客户端连接

·客户端指定套接字地址的连接端口号，在一般情况下，客户端不用指定自己的套接字地址的端口号，当客户调用函数connect进行TCP连接时，系统会自动为它选择一个未用的端口号，当客户端调用函数connect进行TCP连接时，系统会自动为它选择一个未使用的端口号，并且用本地的IP地址来填充套接字地址中的相应项。

·指定客户端的IP地址和连接端口号，表示客户端使用指定的网络设备接口和端口号进行通信

·指定客户端的IP地址，表示客户端使用指定的网络设备接口进行通信，系统自动为客户端选择一个未使用的端口号。一般情况下，只有在主机有多个网络设备接口时使用。

bind函数对应的参数组合方式

C/S IP port 说明

服务器 INADDR\_ANY 非0值 指定服务器的公认端口号

服务器 本地IP地址 非0值 指定服务器的IP地址和公认端口号

客户端 INADDR\_ANY 非0值 指定客户端的连接端口号

客户端 本地IP地址 非0值 指定客户端的IP地址和连接端口号

客户端 本地IP地址 0 指定客户端的IP地址

在编写客户端程序时，通常不适用固定的端口号，除非是在必须使用特定端口的情况下，例如：

·服务器执行主动关闭操作：服务器最后进入TIME\_WAIT状态。当客户机再次与这个服务器进行连接时，仍使用相同的客户机端口号，于是这个连接与前一次连接的套接字对完全相同，这是因为前一次连接处于TIME\_WAIT状态，并未完全消失，所以这次连接请求被拒绝，函数connect以错误返回。

·客户端执行手动关闭操作：客户端最后进入TIME\_WAIT状态，当立刻再次执行这个客户端程序时，客户机将继续与这个固定客户机端口号绑定，但因为前一次连接处于TIME\_WAIT状态，并未消失，系统会发现这个端口号仍被占用，所以这次绑定操作失败，函数bind以错误返回。

Connect函数：

当使用socket函数建立一个套接字并且绑定了地址之后，即可使用connect函数来和服务器建立连接：

int connect(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen); //若连接成功则返回0，否则返回-1

connect函数的参数含义与bind函数相同。

在调用connect函数建立连接之前，客户端应用程序需要指定服务器端进行的套接字地址，而客户端不会指定自己的套接字地址，这时Linux会自动从1024~5000的端口范围中为客户端分配一个未被使用的套接字地址，然后将该端口号和本机IP地址结合在一起放入套接字地址中。

当客户端调用函数connect函数来主动建立连接时，这个函数将启动TCP3次握手过程，在连接建立之后或发生错误时，函数返回。连接过程中可能有以下几种错误情况：

·如果客户端TCP协议没有接收到对它的SYN确认信息，则函数以错误返回，错误类型为ETIMEOUT。

·如果远程TCP协议返回一个RST数据报，则函数立即以错误返回，错误类型为ECONNREFUSED。当在SYN数据报指定的目的端口号没有服务器进程在等待连接时，会发送RST数据报，向客户机报告这个错误。

·如果客户机的SYN数据报导致某个路由器产生“目的地不可达”类型的ICMP消息，则函数以错误返回，错误类型为EHOSTUNREACH或ENETUNREACH。通常情况下，TCP协议在接收到ICMP消息后回记录下这个消息并继续发送几次SYN数据报，如果都宣告失败，函数才会以错误返回。

如果调用函数connect失败，应该用函数close关闭这个套接字，不能再次用这个套接字来调用函数connect。

Listen函数：

int listen(int sockfd, int backlog); //成功则返回0，否则返回-1

参数sockfd为套接字描述符，backlog为设置请求队列的最大长度。该函数的功能有以下两个：

·socket函数创建的套接字是主动套接字，可以用它来进行主动连接（调用connect函数），但不能接受连接请求，而用于服务器的套接字必须能够接收客户端请求。listen函数将一个尚未连接的主动套接字转换成为一个被动套接字，被动套接字可以接收请求。

总结起来就是，若要创建一个倾听套接字，必须首先调用socket函数创建一个主动套接字，然后调用bind函数将套接字与服务器套接字地址绑定在一起，最后调用listen函数进行转换。这3个操作是所有TCP服务器必须执行的操作。

而对于参数backlog的作用，TCP协议为每个倾听套接字维护两个队列：

·未完成连接队列：每个尚未完成3次握手操作的TCP连接在这个队列中占有一项。TCP协议在接收到一个客户机的SYN数据报后，在这个队列中创建一个新条目，然后发送确认消息以及自己的SYN数据报（ACK+SYN），等待客户端对自己的SYN进行确认。此时，套接字处于SYN\_RCVD状态，这个条目保存在队列中，直到客户端返回对自己SYN消息的确认，或者连接超时。

·已完成连接队列：每个已经完成3次握手操作，但尚未被应用程序接收（调用accept函数）的TCP连接在这个队列中占有一项。

参数backlog指定倾听套接字的完成连接对了的最大长度，表示这个套接字能够接收的最大数目的未接收（unaccepted）连接。如果当一个客户端的SYN消息到达，但这个队列已满，TCP协议将会忽略这个SYN，并且不会发送RST数据报。

Accept函数

当服务器倾听到一个连接之后，可以使用函数accept从倾听套接字的完成连接队列中接收一个连接，如果这个完成连接队列为空，则会使得这个进程进入睡眠状态。

int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

当这个函数执行成功时，函数返回值为一个新的套接字描述符，标识这个接收的连接。而且函数还把参数addr指向的结构变量中的地址修改为客户机地址，更新参数addrlen指向的整型变量中存储客户机地址的长度。如果对客户机的地址和长度都不感兴趣，可以将参数addr和addrlen设置为NULL。若函数执行失败，返回-1。

accept函数从倾听套接字的完成连接队列中接收一个已经建立的TCP连接，因为倾听套接字时专为接收客户机请求，完成3次握手的连接操作而用的，所以TCP协议不能使用倾听套接字来标识这个连接，于是TCP协议创建一个新的套接字来标识这个要接收的连接，并将它返回给应用程序。现在有两个套接字：一个是调用accept函数时使用的倾听套接字，另一个是accept函数执行成功返回的连接套接字（connected socket）。这两个套接字的作用是完全不同的：一个服务器进程通常只需要创建一个倾听套接字，在服务器进程的整个活动期间，用它来接收所有客户机连接请求，在服务器进程终止前关闭这个倾听套接字；而对于每个接收的连接，TCP都要创建一个新的连接套接字来标识这个连接，服务器使用这个连接套接字与客户机进行通信，当服务器处理完这个客户机请求时，关闭这个套接字。

Close函数：

当操作完成之后，可以使用close函数来关闭当前建立的连接，这函数定义在头文件<unistd.h>中。

int close(int fd);

close函数将套接字描述符的引用计数减1。如果描述符的引用计数大于0，则表示还有进程引用这个描述符，close函数正常返回；如果描述符的引用计数变为0，则表示再没有进程引用这个描述符，于是启动清除套接字描述符的操作，函数正常返回。清除操作是将这个套接字描述符标记为关闭状态，进程将不再能访问这个套接字，但并不表示套接字已经被删除（联想一下TCP的4次挥手的过程）。TCP将继续使用这个套接字，将尚未发送的数据传递给对方，然后发送FIN数据报，执行关闭操作，直到这个TCP连接完全关闭后，套接字才会被删除。

Read、Write函数：

int read(int fd, char \*buf, int len);

int write(int fd, char \*buf, int len);

fd为套接字描述符；buf指定数据缓冲区；len指定接收或发送的数据量大小（以字节为单位）。函数成功执行时，返回读/写成功的数据量大小，失败则返回-1。

Send、Recv函数：

除了之前说的read和write之外，还可以使用recv和send函数来在套接字中实现数据的发送和接送。这两个函数相比read/write多了一个参数用以指明控制套接字特殊传输方式的各种标志。

ssize\_t send(int sockfd, const void \*buf, size\_t len, int flags);

ssize\_t recv(int sockfd, void \*buf, size\_t len, int flags);

与read/write函数相同，如果调用成功，返回值是代表接收/发送成功的字节数（函数发送的实际长度可能小于其指定的长度），失败则返回-1。flags参数用于指定消息的传送类型，当该值为0时send函数和write函数完全相同，同样recv函数与read函数也相同。

send函数的flags参数除0外还有如下两种取值：

send函数的flags参数取值

MSG\_OOB send函数发送的数据成为带外数据，带外数据是流套接字特有的。在流套接字上传送数据时，数据按它们写出的顺序传送。因为接收进程必须依次读套接字上的当前数据，因此当出现一个紧急情况时，没有办法立即通知接收进程。带外数据正是用于解决这个问题。带外数据在正常的数据流之外发送，其效果相当于越过套接字上所有等待的数据。当它到达接收进程时，接收进程会收到一个信号，从而进程可以立即处理这个数据。

MSG\_DONTROUTE 不在消息中包含路由信息，通常来说普通应用不会关系相应的信息。

recv函数的flags参数除0外还有如下三种取值：

recv函数的flags参数取值

MSG\_OOB 读带外数据

MSG\_PEEK 窥视套接字上的数据而不实际读出它们，即尽管buffer所指对象中填入了所请求的数据，但随后的read或recv将读到相同的数据。

MSG\_WAITALL 请求函数阻塞直至所请求的全部数据都已接收到。

write函数、read函数、send函数和recv函数都是用于TCP下面向连接的套接字数据发送和接收。而在UDP下面向无连接的套接字数据发送和接收则需要使用sendto和recvfrom函数。

Sendto、Recvfrom函数：

ssize\_t sendto(int sockfd, const void \*buf, size\_t len, int flags, const struct sockaddr \*dest\_addr, socklen\_t addrlen);

ssize\_t recvfrom(int sockfd, const void \*buf, size\_t len, int flags, const struct sockaddr \*src\_addr, socklen\_t addrlen);

密码密文处理：利用到两个系统函数

system("stty -icanon");

//设置一次性读完操作，即getchar()不用回车也能获取字符

system("stty -echo");

//关闭回显，即输入任何字符都不显示

为了实现表情功能用到了strcmp函数​

void expression(char name[],char msg[])

{

if(strcmp(msg,"/xl") == 0)

{

sprintf(msg,"表情：%s 做了个笑脸 ☺:-D",name);

}

……

}

为了保存用户数据，如账号密码聊天记录等，用到了sqlite3数据库操作函数

如：

//创建数据库用于保存账号密码

sqlite3 \* Create\_Sqlite(void)

{

sqlite3 \* datebase;

int ret = sqlite3\_open("chat.db", &datebase);

if(ret != SQLITE\_OK)

{

printf("数据库打开失败\n");

return NULL;

}

printf("数据库打开成功\n");

return datebase;

}

//注册时保存用户账号和密码

int Save\_User(Msg \*msg, sqlite3 \*datebase)

{

//打开表

char \*errmsg = NULL;

char buf[100] = "create table if not exists user(NAME TEXT,Password TEXT,Sign\_name TEXT,primary key(NAME))";

int ret = sqlite3\_exec(datebase, buf, NULL, NULL, &errmsg);

if(ret != SQLITE\_OK)

{

printf("打开表失败\n");

printf("sqlite3\_exec: %s\n",errmsg);

return -1;

}

printf("打开表成功\n");

//判断表中是否已存在相同账号名

char \*\*resultp = NULL;

int nrow,ncolumn;

char \*sq1 = "select NAME from user";

ret = sqlite3\_get\_table(datebase, sq1, &resultp, &nrow, &ncolumn, &errmsg);

if(ret != SQLITE\_OK)

{

printf("数据库操作失败\n");

printf("sqlite3\_get\_table: %s\n",errmsg);

return -1;

}

//插入数据

sprintf(buf,"insert into user values('%s', '%s', '你的程序员小可爱已上线')",msg->fromname, msg->password);

ret = sqlite3\_exec(datebase, buf, NULL, NULL,&errmsg);

if(ret != SQLITE\_OK)

{

printf("插入数据失败\n");

printf("sqlite3\_exec: %s\n",errmsg);

return -1;

}

return 0;

}

……

为了整体编译，写了个Makefile

all:client server

client:client.o error.o sqlite3.o

gcc client.o error.o sqlite3.o -o client -lsqlite3 -lpthread

server:sqlite3.o server.o error.o

gcc sqlite3.o server.o error.o -o server -lsqlite3 -lpthread

sqlite3.o:sqlite3.c

gcc -c sqlite3.c

server.o:server.c

gcc -c server.c

client.o:client.c

gcc -c client.c

error.o:error.c

gcc -c error.c

.PHONY:clean

clean:

rm -f \*.o

rm -f server

rm -f client

rm -f \*.db

clear:

rm \*.db

为了直观观察错误类型，写了个error函数

#include"error.h"

//往屏幕输出错误原因

void myerror (char \*str)

{

char \*msg = myStrError ();

printf("%s: %s\n",str,msg);

}

//输入错误原因

char \* myStrError (void)

{

switch (errno)

{

case SOCKET\_INIT :

return "初始化套接字失败.";

case SOCKET\_ACCEPT :

return "链接客户端失败.";

case CREATE\_SQLITE:

return "打开数据库失败.";

case SAVE\_SQLITE:

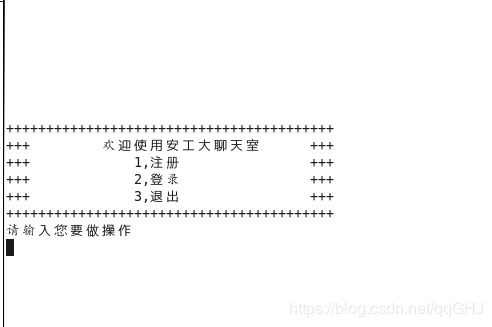
return "数据库插入数据失败,";

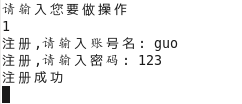
}

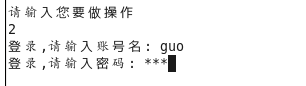
}

1. **系统测试**

登录界面，主要功能是注册，登录，退出（注：图片来自本人csdn博客<https://blog.csdn.net/qqGHJ/article/details/107405985> 下同）

​

注册功能（考虑实用，这里没有密文处理）：​

登录功能（密文处理，密码可删除）：​

**退出功能**：

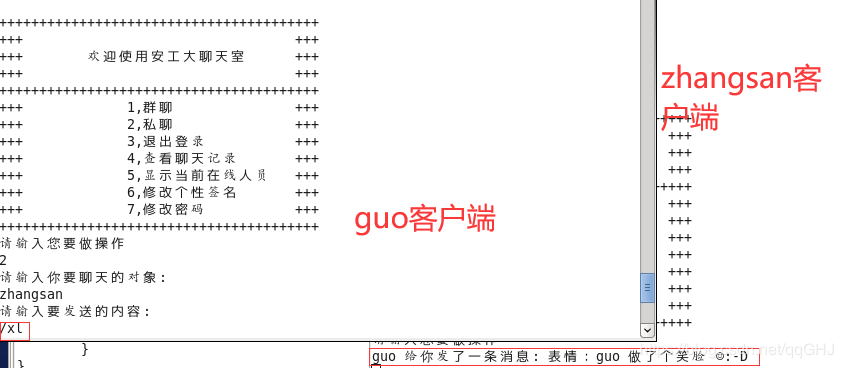
### ​

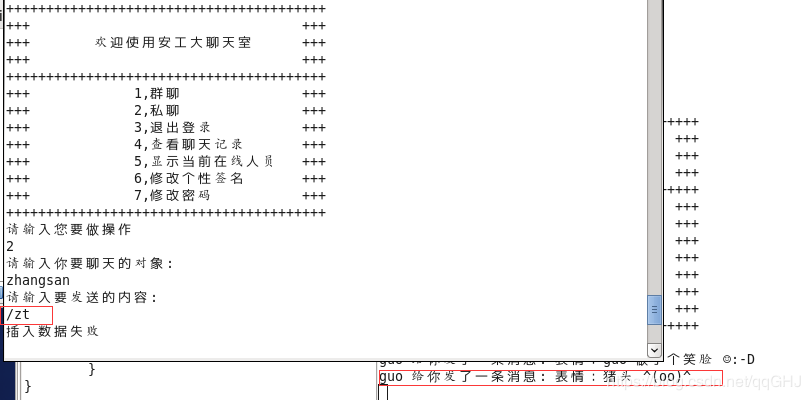
用户主界面：

个性签名功能：

​

表情功能（很简单好玩的功能）：

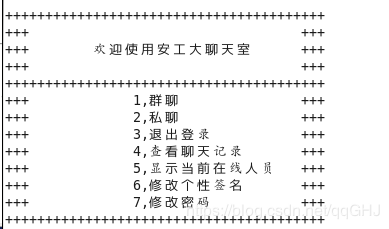
​

​

表情格式为 /+表情缩写，如 /js 表示表情沮丧 /zt 表示表情猪头

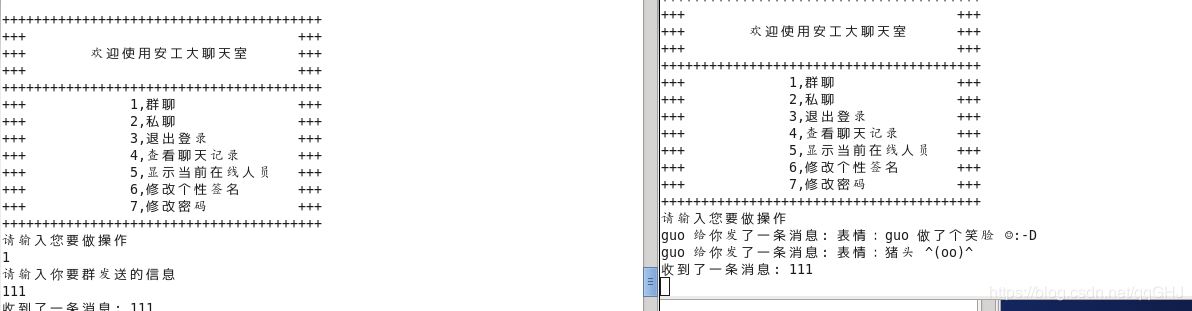
​

功能选项

​

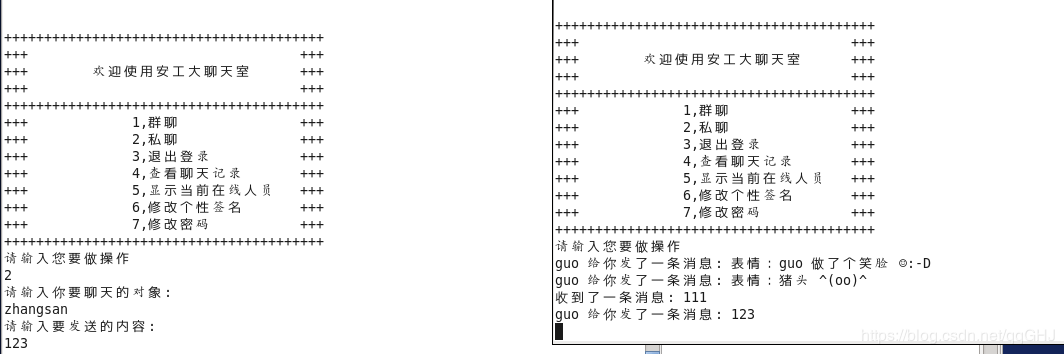
私聊、群聊功能（这里设置一个数组，保存用户在线状态，默认0，每次用户上线为1，群发即对每个状态为1的用户发消息）：

1、群聊

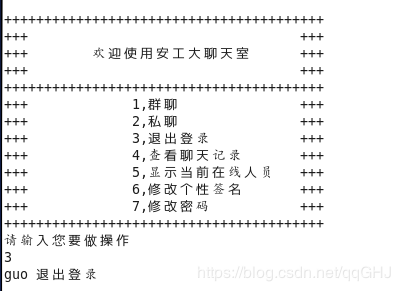


​

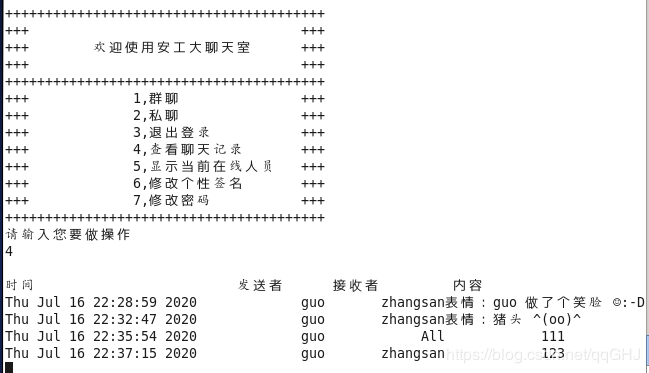
2、私聊：

​

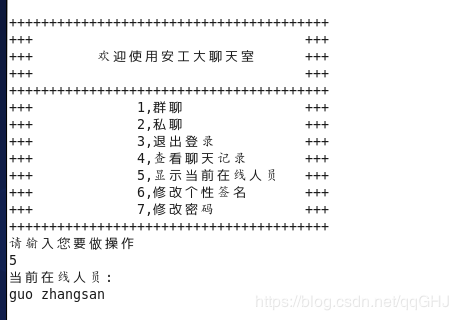
3、退出登录

​

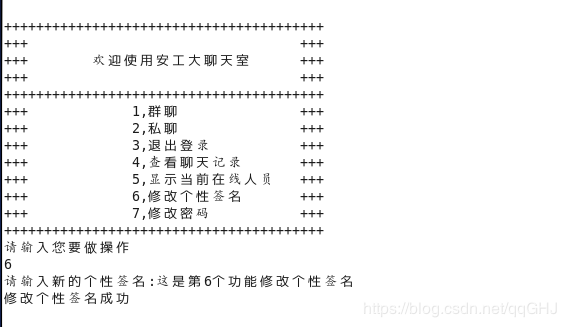
4、查看聊天记录（读sqlite3数据库数据）

​

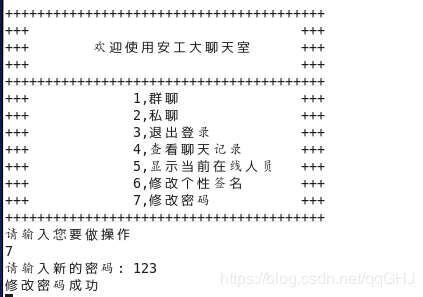
5、查看在线人数：

​

6、修改个性签名

​

7、修改密码（这里为了实用密码也没有密文处理）

​

**五、结束语**

**六、参考文献**

程国钢,张玉兰 《Linux C编程从基础到实践》 清华大学出版社，2015年

# shuaixio CSDN博客 《Linux网络编程-基础与实例》

# china\_jeffery CSDN博客 《网络协议 -- UDP协议（1）介绍》

# AngryChar CSDN博客 《结课项目:聊天室》

# qqGHJ CSDN博客 《计算机网络的基本知识点》

# 百度百科 Linux词条

# 百度百科 聊天工具词条

**七、附录（主体代码）**

**server.c**

#include "server.h"

NowClient user[NOW\_MAX]; //表示当前在线人的资料

//创建监听套接字

int Socket\_init (void)

{

//创建

int listen\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if(listen\_socket == -1)

{

perror("socket error");

printf("套接字创建失败.\n");

return -1;

}

//绑定

struct sockaddr\_in addr;

memset(&addr, 0, sizeof(addr)); //清空内存

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(PORT); //端口和ip要改成大端模式

addr.sin\_addr.s\_addr = htonl(INADDR\_ANY); //INADDR\_ALL表示任意ip

int ret = bind(listen\_socket, (struct sockaddr\*)&(addr), sizeof(addr));

if(ret == -1)

{

perror("bind error");

printf("绑定套接字失败.\n");

return -1;

}

//监听

ret = listen(listen\_socket, 3);

if(ret == -1)

{

perror("listen error");

printf("监听套接字失败.\n");

return -1;

}

printf("等待用户连接.....\n");

return listen\_socket;

}

//链接客户端

int Socket\_Accept (int listen\_socket)

{

struct sockaddr\_in client\_addr;

int len = sizeof(client\_addr);

int client\_socket = accept(listen\_socket,

(struct sockaddr\*)&(client\_addr), &len);

if(client\_socket == -1)

{

perror("accept error");

printf("链接客户端失败\n");

return -1;

}

printf("成功链接了一个客户端 : %s\n"), inet\_ntoa(client\_addr.sin\_addr);

return client\_socket;

}

//服务器创建

int Make\_Server (void)

{

//初始化套接字

int listen\_socket = Socket\_init();

if(listen\_socket == -1)

{

errno = SOCKET\_INIT;

myerror("Socket\_init error");

return -1;

}

while(1)

{

//链接客户端

int client\_socket = Socket\_Accept(listen\_socket);

if(listen\_socket == -1)

{

errno = SOCKET\_ACCEPT;

myerror("Socket\_Accept error");

return -1;

}

//创建进程处理客户端

pthread\_t client\_id;

int ret = pthread\_create(&client\_id, NULL, hanld\_client, (void\*)client\_socket);

if(ret != 0)

{

perror("pthread\_create error");

return -1;

}

pthread\_detach(client\_id); // 线程分离

}

close (listen\_socket);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主界面操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void \* hanld\_client (void\* v) //线程操作客户端

{

int client\_socket = (int)v;

Msg msg;

while(1)

{

//从客户端读取数据

int ret = read(client\_socket, &msg, sizeof(msg));

if (ret == -1)

{

perror ("read error ");

break;

}

// 代表客户端退出

if (ret == 0)

{

printf ("客户端退出\n");

break;

}

switch (msg.cmd)

{

case 1 : // 注册

regis(client\_socket, &msg);

break;

case 2 : //登录

ret = entry(client\_socket, &msg);

write(client\_socket , &msg ,sizeof(Msg));

if (ret == 1)

{

//在线人数加1

int i;

for (i=0; i<NOW\_MAX; i++)

{

if(user[i].socket == 0)

{

strcpy(user[i].name, msg.fromname);

user[i].socket = client\_socket;

printf("客户端在线人数加一\n");

break;

}

}

//用户界面

User\_server(client\_socket, &msg);

}

break;

}

}

}

// 客户端进行注册

int regis(int client\_socket, Msg \*msg)

{

printf(" %s 进行注册.\n",msg->fromname);

//用户账号和密码保存在数据库中

sqlite3 \*datebase = Create\_Sqlite();

if(datebase == NULL)

{

errno = CREATE\_SQLITE;

myerror("Create\_Sqlite");

}

int flag = Save\_User(msg, datebase);

if (flag == -1)

{

errno = SAVE\_SQLITE;

myerror("Save\_User");

msg->cmd = -1;

}

else

{

msg->cmd += 1000;

}

sqlite3\_close(datebase);

write(client\_socket , msg ,sizeof(Msg));

}

//登录账号

int entry(int client\_socket, Msg \*msg)

{

printf(" %s 进行登录.\n",msg->fromname);

//用户登录

sqlite3 \*datebase = Create\_Sqlite();

if(datebase == NULL)

{

errno = CREATE\_SQLITE;

myerror("Create\_Sqlite");

return -1;

}

int flag = Entry\_User(msg, datebase);

if (flag == -1)

{

errno = SAVE\_SQLITE;

myerror("Entry\_User");

return -1;

}

if (flag == -2)

{

printf("登录失败,用户名不存在\n");

msg->cmd = -1;

return -1;

}

if (flag == -3)

{

printf("登录失败,密码错误\n");

msg->cmd = -2;

return -1;

}

else

{

printf("登录成功\n");

msg->cmd += 1000;

return 1;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*用户界面操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//用户界面

void User\_server(int client\_socket, Msg \*msg)

{

int j = 1; //表示循环退出条件

while(j)

{

//从客户端读取数据

int ret = read(client\_socket, msg, sizeof(Msg));

if (ret == -1)

{

perror ("read");

break;

}

// 代表客户端退出

if (ret == 0)

{

//printf ("客户端退出\n");

break;

}

switch (msg->cmd)

{

case 3 : //群聊

server\_chatall(client\_socket, msg);

break;

case 4 : //私聊

server\_chatone(client\_socket, msg);

break;

case 5 : //退出登录

server\_entryout(client\_socket, msg);

j = 0;

break;

case 6 : //显示当前在线人数

see\_nowuser(client\_socket, msg);

break;

case 7 : //修改个性签名

server\_revise\_sign(client\_socket, msg);

break;

case 8 : //修改密码

server\_revise\_password(client\_socket, msg);

break;

}

}

//用户下线

int i ;

for(i=0; i<NOW\_MAX; i++)

{

if(user[i].socket == client\_socket)

{

user[i].socket = 0;

printf("客户端在线人数减一\n");

break;

}

}

}

//群聊

void server\_chatall(int client\_socket, Msg \* msg)

{

printf (" %s 进行群发.\n",msg->fromname);

int i;

for(i=0; i<NOW\_MAX; i++)

{

if (user[i].socket != 0)

{

write(user[i].socket, msg , sizeof(Msg));

}

}

}

//私聊

void server\_chatone(int client\_socket, Msg \* msg)

{

printf ("私聊 %s发送信息给%s\n",msg->fromname,msg->localname);

int i;

for(i=0; i<NOW\_MAX; i++)

{

if(user[i].socket != 0 && strncmp(user[i].name, msg->localname, strlen(msg->localname)) == 0)

{

write(user[i].socket, msg , sizeof(Msg));

printf("私聊成功\n");

break;

}

}

if (i == NOW\_MAX)

{

msg->cmd = -3; //表示私聊失败

write(client\_socket, msg , sizeof(Msg));

printf("私聊失败\n");

}

}

//退出当前登录

void server\_entryout(int client\_socket, Msg \* msg)

{

write(client\_socket, msg , sizeof(Msg));

printf("%s 退出登录\n",msg->fromname);

}

//显示当前在下人数

void see\_nowuser(int client\_socket, Msg \* msg)

{

printf("%s 查看当前在线人员\n",msg->fromname);

int i;

int len;

char buf[1024] = {0};

for(i=0; i<NOW\_MAX; i++)

{

if(user[i].socket != 0)

{

strcat(buf,user[i].name);

len = strlen(buf);

buf[len] = ' ';

}

}

strcpy(msg->msg,buf);

write(client\_socket, msg, sizeof(Msg));

printf("查看成功\n");

}

//修改个性签名

void server\_revise\_sign(int client\_socket, Msg \* msg)

{

printf("%s 修改个性签名\n",msg->fromname);

int ret = revise\_sign\_sqlite(msg); //修改数据库

if (ret == -1)

{

msg->cmd = -7;

printf("%s 修改个性签名失败\n",msg->fromname);

write(client\_socket, msg, sizeof(Msg));

}

printf("%s 修改个性签名成功\n",msg->fromname);

write(client\_socket, msg, sizeof(Msg));

}

//修改密码

void server\_revise\_password(int client\_socket, Msg \* msg)

{

printf("%s 修改密码\n",msg->fromname);

int ret = revise\_password\_sqlite(msg);

if (ret == -1)

{

msg->cmd = -8;

printf("%s 修改密码失败\n",msg->fromname);

write(client\_socket, msg, sizeof(Msg));

}

printf("%s 修改密码成功\n",msg->fromname);

write(client\_socket, msg, sizeof(Msg));

}

int main()

{

Make\_Server ();

return 0;

}

**client.c**

#include "client.h"

char myname[20]; //用于保存本地名字

char signname[40]; //用于保存个性签名

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*网络连接\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void expression(char name[],char msg[])

{

if(strcmp(msg,"/xl") == 0)

{

sprintf(msg,"表情：%s 做了个笑脸 ☺:-D",name);

}

if(strcmp(msg,"/js") == 0)

{

sprintf(msg,"表情：%s 很沮丧 ☹ :-(",name);

}

if(strcmp(msg,"/jy") == 0)

{

sprintf(msg,"表情：%s 很惊讶 :-O",name);

}

if(strcmp(msg,"/hh") == 0)

{

sprintf(msg,"表情：哈哈 ^\_^");

}

if(strcmp(msg,"/kkl") == 0)

{

sprintf(msg,"表情：%s 快哭了 T\_T",name);

}

if(strcmp(msg,"/zk") == 0)

{

sprintf(msg,"表情：%s 抓狂 >\_<",name);

}

if(strcmp(msg,"/mmj") == 0)

{

sprintf(msg,"表情：喵喵叫 (=^\_^=)");

}

if(strcmp(msg,"/yd") == 0)

{

sprintf(msg,"表情：晕 (×\_×)");

}

if(strcmp(msg,"/zt") == 0)

{

sprintf(msg,"表情：猪头 ^(oo)^");

}

if(strcmp(msg,"/dgx") == 0)

{

sprintf(msg,"表情：大狗熊 (￣(工)￣)");

}

if(strcmp(msg,"/bz") == 0)

{

sprintf(msg,"表情：闭嘴吧你 :-x");

}

}

void handle\_password(char \*pass) //用\*代替密码

{

int i=0;

system("stty -icanon"); //设置一次性读完操作，即getchar()不用回车也能获取字符

system("stty -echo"); //关闭回显，即输入任何字符都不显示

while(i < 16)

{

pass[i]=getchar(); //获取键盘的值到数组中

if(i == 0 && pass[i] == BACKSPACE)

{

i=0; //若开始没有值，输入删除，则，不算值

pass[i]='\0';

continue;

}

else if(pass[i] == BACKSPACE)

{

printf("\b \b");//若删除，则光标前移，输空格覆盖，再光标前移

pass[i]='\0';

i=i-1; //返回到前一个值继续输入

continue; //结束当前循环

}

else if(pass[i] == '\n') //若按回车则，输入结束

{

pass[i]='\0';

break;

}

else

{

printf("\*");

}

i++;

}

system("stty echo"); //开启回显

system("stty icanon"); //关闭一次性读完操作，即getchar()必须回车也能获取字符

}

//连接服务器

int Connect\_Server(void)

{

//创建套接字

int client\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if(client\_socket == -1)

{

perror("socket error");

return -1;

}

//链接服务器

struct sockaddr\_in addr;

memset(&addr, 0, sizeof(addr));

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(PORT);

inet\_aton("127.0.0.1",&(addr.sin\_addr));

int ret = connect(client\_socket,(struct sockaddr \*)&addr, sizeof(addr));

if(ret == -1)

{

perror("connect error");

return -1;

}

printf("成功连接到服务器 : %s\n",inet\_ntoa(addr.sin\_addr));

return client\_socket;

}

//访问服务器 主界面

int Ask\_server(int client\_socket)

{

char ch;

int ret;

while(1)

{

main\_menu();

printf("请输入您要做操作\n");

scanf("%c",&ch);

while(getchar() != '\n');

switch(ch)

{

case '1': //注册

regis(client\_socket);

break;

case '2': //登录

ret = entry(client\_socket);

if (ret == 1)

{

User\_used(client\_socket); //调用函数表示用户界面

}

break;

case '3': //退出

exit(0);

break;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*界面\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//客户端主界面

void main\_menu(void)

{

system("clear");

printf("\n\n\n\n\n\n\n\n");

printf("+++++++++++++++++++++++++++++++++++++++++\n");

printf("+++\t 欢迎使用安工大聊天室 +++\n");

printf("+++\t\t1,注册 +++\n");

printf("+++\t\t2,登录 +++\n");

printf("+++\t\t3,退出 +++\n");

printf("+++++++++++++++++++++++++++++++++++++++++\n");

}

//用户登录界面

void user\_menu(void)

{

system("clear");

printf("%s: %s\n",myname,signname);

printf("\n\n\n\n\n\n\n");

printf("++++++++++++++++++++++++++++++++++++++++\n");

printf("+++ +++\n");

printf("+++\t 欢迎使用安工大聊天室 +++\n");

printf("+++ +++\n");

printf("++++++++++++++++++++++++++++++++++++++++\n");

printf("+++\t\t1,群聊 +++\n");

printf("+++\t\t2,私聊 +++\n");

printf("+++\t\t3,退出登录 +++\n");

printf("+++\t\t4,查看聊天记录 +++\n");

printf("+++\t\t5,显示当前在线人员 +++\n");

printf("+++\t\t6,修改个性签名 +++\n");

printf("+++\t\t7,修改密码 +++\n");

printf("++++++++++++++++++++++++++++++++++++++++\n");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主界面功能\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//注册账号

void regis(int client\_socket)

{

Msg msg;

msg.cmd = 1;

printf("注册,请输入账号名: ");

scanf("%s",msg.fromname);

while(getchar() != '\n');

printf("注册,请输入密码: ");

scanf("%s",msg.password);

while(getchar() != '\n');

write(client\_socket, &msg, sizeof(msg));

read(client\_socket, &msg, sizeof(msg));

if(msg.cmd == 1001)

{

printf("注册成功\n");

}

else if (msg.cmd == -1)

{

printf("用户名已存在,注册失败\n");

}

else

{

printf("系统繁忙,注册失败\n");

}

sleep(2);

}

//登录账号

int entry(int client\_socket)

{

Msg msg;

msg.cmd = 2;

int i\_password=0;

char \*ch\_password;//用于暂存密码

printf("登录,请输入账号名: ");

scanf("%s",msg.fromname);

while(getchar() != '\n');

printf("登录,请输入密码: ");

/\*

//ch\_password=getchar(); //读取字符不显示

//putchar('\*');

//while(ch\_password= '\n')

{

//msg.password[++i\_password]=ch\_password;

//ch\_password=getchar(); //读取字符不显示

//system("stty -echo");

//putchar('\*');

ch\_password=getpass("Input a password");

strcpy(msg.password,ch\_password);

}

\*/

handle\_password(msg.password);

//scanf("%s",msg.password);

//while(getchar() != '\n');

printf("\n");

write(client\_socket, &msg, sizeof(msg));

read(client\_socket, &msg, sizeof(msg));

if(msg.cmd == -1) //表示用户不存在

{

printf("登录失败,用户不存在.\n");

sleep(2);

return -1;

}

if(msg.cmd == -2)

{

printf("登录失败,密码错误.\n");

sleep(2);

return -2;

}

if(msg.cmd == 1002)

{

printf("登录成功,登录中...\n");

strcpy(myname,msg.fromname); //保存在线名字

strcpy(signname, msg.signname); //保存个性签名

sleep(2);

return 1;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*用户界面功能\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//用户界面

void User\_used(int client\_socket)

{

//要进行读写分离

pthread\_t read\_id;

pthread\_create(&read\_id, NULL, readMsg, (void \*)client\_socket);

pthread\_detach(read\_id); //等待线程分离

char ch;

int i = 1;

while(i)

{

user\_menu();

printf("请输入您要做操作\n");

scanf("%c",&ch);

while(getchar() != '\n');

switch(ch)

{

case '1': //群聊

chat\_all(client\_socket);

break;

case '2': //私聊

chat\_one(client\_socket);

break;

case '3': //退出登录

entry\_out(client\_socket);

i = 0;

break;

case '4': //查看聊天记录

look\_chat();

break;

case '5': //显示当前在线人数

see\_now\_time(client\_socket);

break;

case '6': //修改个性签名

revise\_sign(client\_socket);

break;

case '7': //修改密码

revise\_password(client\_socket);

break;

}

}

}

//读写分离

void \* readMsg (void \*v)

{

int client\_socket = (int)v;

Msg buf;

int i = 0;

while(1)

{

//Msg buf;

bzero(&buf,sizeof(buf));

int ret = read(client\_socket, &buf, sizeof(Msg));

if(ret == -1)

{

perror("read");

break;

}

switch(buf.cmd)

{

case 3: //群聊

expression(buf.fromname,buf.msg);

printf("收到了一条消息: %s\n",buf.msg);

save\_Chat(&buf); //保存聊天记录

break;

case 4: //私聊

expression(buf.fromname,buf.msg);

printf("%s 给你发了一条消息: %s\n",buf.fromname,buf.msg);

save\_Chat(&buf); //保存聊天记录

break;

case -3: //私聊失败

printf("私聊失败,用户不存在或下线\n");

break;

case 5 : //退出登录

printf("%s 退出登录\n",buf.fromname);

sleep(1);

pthread\_exit(NULL); //线程退出

break;

case 6 : //显示当前在线人数

printf("当前在线人员:\n");

printf("%s\n",buf.msg);

break;

case 7 : //修改个性签名成功

strcpy(signname,buf.signname);

printf("修改个性签名成功\n");

break;

case -7 : //修改个性签名失败

printf("修改个性签名失败\n");

break;

case 8 : //修改密码成功

printf("修改密码成功\n");

break;

case -8 : //修改密码失败

printf("修改密码失败\n");

break;

default:break;

}

}

}

//群聊

void chat\_all(int client\_socket)

{

Msg msg;

msg.cmd = 3;

strcpy(msg.fromname,myname);

strcpy(msg.localname,"All");

printf("请输入你要群发送的信息\n");

scanf("%s",msg.msg);

while(getchar() != '\n');

write(client\_socket, &msg, sizeof(Msg));

sleep (2);

}

//私聊

void chat\_one(int client\_socket)

{

Msg msg;

msg.cmd = 4;

printf("请输入你要聊天的对象:\n");

scanf ("%s",msg.localname);

while(getchar() != '\n');

printf("请输入要发送的内容: \n");

scanf("%s",msg.msg);

while(getchar() != '\n');

strcpy (msg.fromname,myname);

write(client\_socket, &msg, sizeof(Msg));

save\_Chat(&msg); //保存聊天记录

sleep(2);

}

//退出登录

void entry\_out(int client\_socket)

{

Msg msg;

msg.cmd = 5;

strcpy (msg.fromname,myname);

write(client\_socket, &msg, sizeof(Msg));

}

//查看聊天记录

void look\_chat(void)

{

Msg msg;

strcpy(msg.fromname,myname);

see\_chat(&msg);

}

//显示当前在线人数

void see\_now\_time(int client\_socket)

{

Msg msg;

msg.cmd = 6;

strcpy (msg.fromname,myname);

write(client\_socket, &msg, sizeof(Msg));

sleep(2);

}

//修改个性签名

void revise\_sign(int client\_socket)

{

Msg msg;

msg.cmd = 7;

printf("请输入新的个性签名:");

scanf("%s",msg.signname);

while(getchar() != '\n');

strcpy(msg.fromname,myname); //需要保存名字

write(client\_socket, &msg, sizeof(Msg));

sleep(2);

}

//修改密码

void revise\_password(int client\_socket)

{

Msg msg;

msg.cmd = 8;

printf("请输入新的密码: ");

scanf("%s",msg.password);

while(getchar() != '\n');

strcpy(msg.fromname,myname); //需要保存名字

write(client\_socket, &msg, sizeof(Msg));

sleep(2);

}

int main()

{

int client\_socket = Connect\_Server();

Ask\_server(client\_socket);

close(client\_socket);

return 0;

}

**sqlite3.c**

#include "network.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*服务器数据库\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//创建数据库用于保存账号密码

sqlite3 \* Create\_Sqlite(void)

{

sqlite3 \* datebase;

int ret = sqlite3\_open("chat.db", &datebase);

if(ret != SQLITE\_OK)

{

printf("数据库打开失败\n");

return NULL;

}

printf("数据库打开成功\n");

return datebase;

}

//注册时保存用户账号和密码

int Save\_User(Msg \*msg, sqlite3 \*datebase)

{

//打开表

char \*errmsg = NULL;

char buf[100] = "create table if not exists user(NAME TEXT,Password TEXT,Sign\_name TEXT,primary key(NAME))";

int ret = sqlite3\_exec(datebase, buf, NULL, NULL, &errmsg);

if(ret != SQLITE\_OK)

{

printf("打开表失败\n");

printf("sqlite3\_exec: %s\n",errmsg);

return -1;

}

printf("打开表成功\n");

//判断表中是否已存在相同账号名

char \*\*resultp = NULL;

int nrow,ncolumn;

char \*sq1 = "select NAME from user";

ret = sqlite3\_get\_table(datebase, sq1, &resultp, &nrow, &ncolumn, &errmsg);

if(ret != SQLITE\_OK)

{

printf("数据库操作失败\n");

printf("sqlite3\_get\_table: %s\n",errmsg);

return -1;

}

//插入数据

sprintf(buf,"insert into user values('%s', '%s', '你的程序员小可爱已上线')",msg->fromname, msg->password);

ret = sqlite3\_exec(datebase, buf, NULL, NULL,&errmsg);

if(ret != SQLITE\_OK)

{

printf("插入数据失败\n");

printf("sqlite3\_exec: %s\n",errmsg);

return -1;

}

return 0;

}

//登录时检查用户账号和密码

int Entry\_User(Msg \*msg, sqlite3 \*datebase)

{

//打开表

char \*errmsg = NULL;

char buf[100] = "create table if not exists user(NAME TEXT,Password TEXT,Sign\_name TEXT,primary key(NAME))";

int ret = sqlite3\_exec(datebase, buf, NULL, NULL, &errmsg);

if(ret != SQLITE\_OK)

{

printf("打开表失败\n");

printf("sqlite3\_exec: %s\n",errmsg);

return -1;

}

printf("打开表成功\n");

//判断账号密码

char \*\*resultp = NULL;

int nrow,ncolumn;

char \*sq1 = "select \* from user";

ret = sqlite3\_get\_table(datebase, sq1, &resultp, &nrow, &ncolumn, &errmsg);

if(ret != SQLITE\_OK)

{

printf("数据库操作失败\n");

printf("sqlite3\_get\_table: %s\n",errmsg);

return -1;

}

int i;

int j = 0 ; //表示密码不对

int flag = 0; //表示账号不存在

for(i=3; i<(nrow+1)\*ncolumn ;i+=3)

{

if(strcmp(resultp[i], msg->fromname) == 0)

{

if (strcmp(resultp[i+1], msg->password) != 0)

{

j = 1;

}

flag = 1;

break;

}

}

ret = 0;

if (flag != 1)

{

ret = -2; //表示名字不存在

}

else if (j == 1)

{

ret = -3; //表示密码不对

}

else

strcpy(msg->signname,resultp[i+2]); //保存个性签名

sqlite3\_free\_table(resultp); //释放空间

return ret; //表示登录成功

}

//修改数据库 (个性签名)

int revise\_sign\_sqlite(Msg \* msg)

{

sqlite3 \* datebase = Create\_Sqlite();

char \*errmsg = NULL;

char buf[200];

sprintf(buf,"update user set Sign\_name = '%s' where Name = '%s'",msg->signname,msg->fromname);

int ret = sqlite3\_exec(datebase, buf, NULL, NULL,&errmsg);

if(ret != SQLITE\_OK)

{

printf("修改数据失败\n");

printf("sqlite3\_exec: %s\n",errmsg);

sqlite3\_close(datebase); //关闭表

return -1;

}

printf("修改数据成功\n");

sqlite3\_close(datebase);

return 0;

}

//修改密码(数据库)

int revise\_password\_sqlite(Msg \* msg)

{

sqlite3 \* datebase = Create\_Sqlite();

char \*errmsg = NULL;

char buf[200];

sprintf(buf,"update user set Password = '%s' where Name = '%s'",msg->password,msg->fromname);

int ret = sqlite3\_exec(datebase, buf, NULL, NULL,&errmsg);

if(ret != SQLITE\_OK)

{

printf("修改数据失败\n");

printf("sqlite3\_exec: %s\n",errmsg);

sqlite3\_close(datebase); //关闭表

return -1;

}

printf("修改数据成功\n");

sqlite3\_close(datebase);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*客户端数据库\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//创建数据库用于保存聊天记录

sqlite3 \* Create\_user\_sqlite(Msg \* msg)

{

sqlite3 \* datebase;

char name[23];

sprintf(name ,"%s.db",msg->fromname);

int ret = sqlite3\_open(name, &datebase);

return datebase;

}

//保存聊天记录到数据库中

void save\_Chat(Msg \*msg)

{

sqlite3 \* datebase = Create\_user\_sqlite(msg);

//打开表

char \*errmsg = NULL;

char buf[200] = "create table if not exists chat(时间 TEXT,发送者 TEXT,接收者 TEXT,内容 TEXT)";

int ret = sqlite3\_exec(datebase, buf, NULL, NULL, &errmsg);

if(ret != SQLITE\_OK)

{

printf("打开表失败\n");

return ;

}

time\_t t;

t = time(&t);

char time[100];

strcpy(time,ctime(&t));

int len = strlen(time);

time[len-1] = '\0'; //去掉回车

sprintf(buf,"insert into chat values('%s','%s','%s','%s')",time,msg->fromname,msg->localname,msg->msg);

ret = sqlite3\_exec(datebase, buf, NULL, NULL,&errmsg);

if(ret != SQLITE\_OK)

{

printf("插入数据失败\n");

return ;

}

sqlite3\_close(datebase);

}

//查看聊天记录(调用数据库)

void see\_chat(Msg \* msg)

{

sqlite3 \* datebase = Create\_user\_sqlite(msg);

//打开表

char \*errmsg = NULL;

char buf[200] = "create table if not exists chat(时间 TEXT,发送者 TEXT,接收者 TEXT,内容 TEXT)";

int ret = sqlite3\_exec(datebase, buf, NULL, NULL, &errmsg);

if(ret != SQLITE\_OK)

{

printf("打开表失败\n");

return ;

}

//查看聊天记录

char \*\*resultp = NULL;

int nrow,ncolumn;

char \*sq1 = "select \* from chat";

ret = sqlite3\_get\_table(datebase, sq1, &resultp, &nrow, &ncolumn, &errmsg);

if(ret != SQLITE\_OK)

{

printf("数据库操作失败\n");

printf("sqlite3\_get\_table: %s\n",errmsg);

return ;

}

int i;

for(i=0; i<(nrow+1)\*ncolumn ;i++)

{

if(i%4 == 0)

{

printf("\n");

printf("%-25s",resultp[i]);

}

else

printf("%15s",resultp[i]);

}

printf("\n");

sqlite3\_free\_table(resultp); //释放空间

sqlite3\_close(datebase);

sleep(5);

}