一、课程设计的目的和意义

计算机网络课程设计的目的，是为了让学生更深入地掌握计算机网络的核心内容，实现理论与实践相结合。让学生用具体的实践成果，体现对理论知识的掌握程度。有利于学生提高计算机网络的实践能力，加深对计算机网络理论知识的理解。其基本目的是：

1．培养学生理论联系实际的设计思想，训练综合运用所学的基础理论知识，结合生产实际分析和解决网络应用中问题的能力，从而使基础理论知识得到巩固和加深。

2．学习掌握网络应用工程的一般设计过程和方法。

二、设计题目和要求

2.1 选题

基于tcp协议的简易聊天室

2.2 设计目的

1.了解socket工作流程，以及通信原理

2.了解mysql在C语言中的调用方法

3.了解多线程程序运行逻辑

2.3 课程设计系统组成及模块功能

此课程设计实现了基于tcp的客户/服务器通信程序，需要实现以下一些基本功能：

1.通过客户端实现用户的注册以及登录。

2.服务端在收到用户注册或登录信息后进行处理。

3.实现多用户同时登录的聊天室

4.用户可以查询当前聊天室的在线状态

三、设计内容

3.1 MySQL增删改查功能实现

3.1.1 头文件定义

```
#ifndef _MYSQL_CONNECT_
#define _MYSQL_CONNECT_

#include <mysql/mysql.h>
#include <stdio.h>
#include <stdbool.h>

// 创建用户信息结构体
typedef struct
{
    char username[20];
    char userpasswd[20];
} User;

// 创建用户在线信息
typedef struct
{
    char username[20];
    char userstate[20];
} Onlineuser;

// 定义一个结构体，用于存储MySQL连接的相关信息
typedef struct mysql_conn
{
    MYSQL *condb; // 定义一个MySQL连接指针
    char *hostIP; // 定义一个字符串指针，用于存储主机名
    char *user;   // 定义一个字符串指针，用于存储用户名
    char *passwd; // 定义一个字符串指针，用于存储密码
    char *db;     // 定义一个字符串指针，用于存储数据库名
} mysql_conn;

// 功能实现-----------------------------------------

// 初始化数据库
// 返回值：MYSQL指针
// 参数：MYSQL指针
MYSQL *db_init(MYSQL *condb);

// 连接数据库
// 返回值：MYSQL指针，地址，用户名，密码，数据库名称
// 参数：MYSQL指针
MYSQL *db_connect(MYSQL *condb, char *hostIP, char *username, char *passwd, char
*db);
```

```
// 显示表信息
// 返回值：bool
// 参数：MYSQL指针，SQL查询字符串
bool info_table(MYSQL *condb, char *query);

// 判断表中name字段是否存在
// 返回值：bool
// 参数：MYSQL指针，用户名
bool if_name_exist(MYSQL *condb, char *name);

// 判断用户名与密码是否匹配
// 返回值：  bool
// 参数：MYSQL指针，用户名，密码
bool judge_user(MYSQL *condb, char *name, char *passwd);

// 用户注册
// 返回值：bool
// 参数：MYSQL指针，用户名，密码
bool insert_user(MYSQL *condb, char *name, char *passwd);

// 用户注销
// 返回值：bool
// 参数：MYSQL指针，用户名，密码
bool drop_user(MYSQL *condb, char *name);

// 功能：用户上线
// 参数：MYSQL指针，用户名，用户在线状态
// 返回值：空
bool insert_user_online(MYSQL *condb, char *name, char *state);

// 功能：用户下线
// 参数：con，所需删除用户的用户名
bool drop_user_online(MYSQL *con, char *name);

// 实现结束----------------------------------------

#endif
```

3.1.2 操作实现

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include<stdbool.h>
    #include <mysql/mysql.h>
    #include "../erro/erro.h"
    #include "mysql_connect.h"

    //定义一个用于判断函数返回值的变量
    int ret;

    //初始化数据库
    //返回值：MYSQL指针
    //参数：MYSQL指针
```

```
    MYSQL* db_init(MYSQL *condb){
        //在多线程环境中需要调用这个函数的
        ret = mysql_library_init(0,NULL,NULL);
        if(ret != 0){
            info_erro("could not initialize MYSQL library");
            exit(1);
        }
        condb = mysql_init(NULL);
        return condb;
    }

    //连接数据库
    //返回值：MYSQL指针，地址，用户名，密码，数据库名称
    //参数：MYSQL指针
    MYSQL* db_connect(MYSQL* condb,char *hostIP,char* username,char*
passwd,char* db){
        if(!(mysql_real_connect(condb,hostIP,username,passwd,db,3306,NULL,0))){
            mysql_error(condb);
            info_erro("连接错误\n可能是网络问题");//判断连接是否错误
            exit(1);
        }
        printf("连接成功！\n");
        return condb;
    }

    //显示表信息
    //返回值：bool 1：成功 0：失败
    //参数：MYSQL指针，SQL查询字符串
    bool info_table(MYSQL* condb,char* query){
        MYSQL_RES *mysql_res; //指向结果集对象
        MYSQL_ROW *mysql_row; //指向结果集的一行
        MYSQL_FIELD *field;    //指向结果集的一个字段
        int i,j;//判断循环变量
        int num_row,num_col;//结果集中的行列
        //char display[100][20];//用于保存输出表的内容
        //执行查询的语句
        ret = mysql_real_query(condb,query,strlen(query));
        if(ret != 0){
            info_erro("查询失败！");
            return false;
        }
        //保存查询结果
        if((mysql_res = mysql_store_result(condb)) == NULL){
            //将查询结果集保存到mysql_res，如果失败返回NULL
            info_erro("结果保存失败！");
            return false;
        }
        num_row = mysql_num_rows(mysql_res);//获取行数2
        num_col = mysql_num_fields(mysql_res);//获取取列数3
        printf("=======================================\n");
        int k=0; // 记录数组的字段
        for( i=0;i<num_row;i++){
            //遍历每一行
            if((mysql_row = mysql_fetch_row(mysql_res)) == NULL){
                break;//获取一行的数据，如果返回NULL则说明没有数据了
```

```
            }
            if( i == 0){
                //如果是表头，打印表头
                while(field = mysql_fetch_field(mysql_res)){
                    //获取每个字段的信息，如果返回空指针，表示没有字段了
                    printf("%-10s",field->name);//打印字段名
                    //strcpy(display[k],field->name);//将字段名写入display数组
                    //strcpy(display[0],field->name);
                    k++;
                }
                printf("\n");
            }
            mysql_fetch_lengths(mysql_res);//获取每个字段的长度
            for( j = 0;j<num_col;j++){
                printf("%-10s",mysql_row[j]);
                //strcpy(display[k2],mysql_row[j]);
            }
            printf("\n");
        }
        printf("=======================================\n");
        mysql_free_result(mysql_res);
        return true;
    }

    //判断表中name字段是否存在
    //返回值：bool 1：存在 0：不存在
    //参数：MYSQL指针，用户名
    bool if_name_exist(MYSQL* condb,char* name){
        User user1;
        char query[1024];
        strcpy(user1.username,name);

        MYSQL_RES *res;
        MYSQL_ROW row;
        MYSQL_FIELD *field;

        int num_fields;//定义结果集列数
        //拼接查询语句
        sprintf(query,"select *from user where name = '%s'",user1.username);
        mysql_query(condb,query);//执行语句
        res = mysql_store_result(condb); //保存查询结果
        num_fields = mysql_num_fields(res);
        while((row = mysql_fetch_row(res))){
            //遍历每一行数据
            if(field = mysql_fetch_field(res)){
                //获取字段信息
                if(!strcmp(user1.username,row[1])){//strcmp匹配成功返回1
                    return true;//不能使用user1.username == row[1]
                }                                  //这比较的只是指针
            }
        }
        mysql_free_result(res);
        mysql_commit(condb); //提交事务
        return false;
    }
```

```
    //判断用户名与密码是否匹配
    //返回值： bool 1：匹配 0：不匹配
    //参数：MYSQL指针，用户名，密码
    bool judge_user(MYSQL* condb,char* name,char* passwd){
        User user1;
        char query[1024];
        strcpy(user1.username,name);
        strcpy(user1.userpasswd,passwd);
        MYSQL_RES *res;
        MYSQL_ROW row;
        MYSQL_FIELD *field;

        int num_fields;//定义结果集列数
        //拼接查询语句
        sprintf(query,"select *from user where name = '%s'",user1.username);
        mysql_query(condb,query);//执行语句
        res = mysql_store_result(condb); //保存查询结果
        num_fields = mysql_num_fields(res);

        while((row = mysql_fetch_row(res))){
            if(field = mysql_fetch_field(res)){
                //strcmp如果匹配的返回值0，则使用!
                if(!strcmp(user1.username,row[1]) && !strcmp(user1.userpasswd
,row[2])){
                    printf("登陆成功\n");
                    return true;
                }
            }
        }
        mysql_free_result(res);
        mysql_commit(condb);
        return false;
    }

    //用户注册
    //返回值：bool
    //参数：MYSQL指针，用户名，密码
    bool insert_user(MYSQL* condb,char* name,char* passwd){
        User user1;
        char query[1024];

        MYSQL_RES *res;
        strcpy(user1.username,name);
        strcpy(user1.userpasswd,passwd);
        //判断用户是否存在
        if(if_name_exist(condb,name)){
            printf("用户存在，请登陆");
            return false;
        }
        sprintf(query,"insert into user (name,passwd) values('%s','%s')",
                        user1.username,user1.userpasswd);
        //执行查询的语句
        ret = mysql_real_query(condb,query,strlen(query));
        if(ret != 0){
```

```
            mysql_rollback(condb);//回转事物
            info_erro("注册失败！");
            return false;
        }
        mysql_commit(condb);
        return true;
    }

    //用户注销
    //返回值：bool
    //参数：MYSQL指针，用户名，密码
    bool drop_user(MYSQL* condb,char* name){
        char query[1024];
        if(!if_name_exist(condb,name)){
            info_erro("注销失败！请重新检查用户名\n");
            return false;
        }
        sprintf(query,"delete from user where name = '%s'",name);
        ret = mysql_real_query(condb,query,strlen(query));
        if(ret != 0){    if(!if_name_exist(condb,name)){
            info_erro("注销失败！请重新检查用户名\n");
            return false;
        }
            mysql_rollback(condb);//回转事物
            info_erro("注销失败！请重新检查用户名\n");
            return false;
        }
        mysql_commit(condb);
            if( mysql_affected_rows(condb)>0){
                printf("删除成功\n");
            }
        return true;
    }

    //功能：用户上线
//参数：MYSQL指针，用户名，用户在线状态
//返回值：空
bool insert_user_online(MYSQL *condb,char *name,char *state){
    Onlineuser user1;
    char query[1024];
    strcpy(user1.username,name);
    strcpy(user1.userstate,state);
    sprintf(query,"insert into user_online(name,state) values('%s','%s')",
        user1.username ,user1.userstate);
    ret = mysql_real_query(condb,query,strlen(query));
    if(ret != 0){
            mysql_rollback(condb);//回转事物
            info_erro("注册失败！");
            return false;
    }
    mysql_commit(condb);
    return true;
}

//功能：用户下线
```

```
//参数：con，所需删除用户的用户名
bool drop_user_online(MYSQL *condb,char *name){
    char query[1024];
    sprintf(query,"delete from user_online where name = '%s'",name);
    ret = mysql_real_query(condb,query,strlen(query));
    if(ret != 0){
        mysql_rollback(condb);//回转事物
        info_erro("注销失败！请重新检查用户名\n");
        return false;
    }
    mysql_commit(condb);
    if( mysql_affected_rows(condb)>0){
        printf("删除成功\n");
    }
    return true;
}

    //实现结束----------------------------------------
    /*
    //功能测试代码
       int main(){
        MYSQL *mysql_handle = NULL;
        int m,n;
        int len = 0;
        mysql_handle = db_init(mysql_handle);
        //连接测试
        mysql_handle =
db_connect(mysql_handle,"47.120.41.232","root","123456","chat");
        //表查询测试
        info_table(mysql_handle,"select *from user");
        printf("\n*****************************\n");
        //插入测试
        insert_user(mysql_handle,"sfen","12");
        info_table(mysql_handle,"select *from user");
        printf("\n*****************************\n");
        //重复name测试
        if(if_name_exist(mysql_handle,"chen")){
            printf("该 用户存在\n");
        }else{
            printf("该用户不存在\n");
        }
        printf("\n*****************************\n");
        //name passwd不匹配测试
        if(judge_user(mysql_handle,"chen","12")){
            printf("匹配\n");
        }else{
            printf("不匹配\n");
        }
        printf("\n*****************************\n");
        //删除测试
        if(drop_user(mysql_handle,"sfen")){
            printf("注销成功\n");
        }else{
            printf("注销失败\n");
        }
```

```
        info_table(mysql_handle,"select *from user");
        printf("\n*******************************\n");
        printf("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$\n");
        //在线用户插入测试
        insert_user_online(mysql_handle,"zhang","online");
        info_table(mysql_handle,"select *from user_online");
        printf("\n*******************************\n");
        //在线用户删除测试
        drop_user_online(mysql_handle,"zhang");
        info_table(mysql_handle,"select *from user_online");
        printf("\n*******************************\n");
        mysql_close(mysql_handle);
        mysql_library_end();
        return 0;
        }

    */
   //`mysql_config --cflags --libs`
   // gcc mysql_connect.c ../erro/erro.c -o test `mysql_config --cflags --libs`
```

3.2 服务端登录注册功能实现

3.2.1 头文件定义

```
#ifndef _SERVRELOGIN_
#define _SERVERLOGIN_

#include <mysql/mysql.h>
#define MAXLINE 1024

typedef struct{
    char online[303][20]; // 在线用户
    int len; // 用户数量
    char msg[1024];//消息内容
    char name[20];//用户账号
    char passwd[20];//用户密码
    int cmd;//消息类型
}Messges;

//注册
void Reg(int client_socket, Messges msg,MYSQL *conndb);
//登录
void Entry(int client_socket,Messges msg,MYSQL *condb);

#endif
```

3.2.2 操作实现

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <stdlib.h>
```

```c
#include <mysql/mysql.h>
#include "../mysql_connection/mysql_connect.h"
#include "serverlogin.h"

//注册
void Reg(int client_socket, Messges msg,MYSQL *conndb){
    //检测用户名是否存在
    if(if_name_exist(conndb,msg.name)){
            msg.cmd = -3;//用户存在，注册失败
            return 0;
    }
    printf("用户%s开始注册...\n",msg.name);
    //插入到数据库中
    if(insert_user(conndb,msg.name,msg.passwd)){
        msg.cmd = 101;//注册成功状态码
        insert_user_online(conndb,msg.name,"online");
    }else{
        msg.cmd = -1;//注册失败
    }
    write(client_socket,&msg,sizeof(msg));
}
//登录
void Entry(int client_socket,Messges msg,MYSQL *conndb){
    //检测用户名与密码是否正确
    if(!judge_user(conndb,msg.name,msg.passwd)){
        msg.cmd = -1;//登陆失败
        //检测用户名是否存在
        if(!if_name_exist(conndb,msg.name)){
            msg.cmd = -2;//登陆失败
        }
    }else{
        drop_user_online(conndb,msg.name);
        insert_user_online(conndb,msg.name,"online");
        msg.cmd = 102;
    }
    write(client_socket,&msg,sizeof(msg));
}
```

3.3 客户端登录注册功能实现

3.3.1 头文件定义

```c
#ifndef _CLIENTLOGIN_
#define _CLIENTLOGIN_

typedef struct{
    char online[303][20]; // 在线用户
    int len; // 用户数量
    char msg[1024];//消息内容
    char name[20];//用户账号
    char passwd[20];//用户密码
    int cmd;//消息类型
}Messges;
```

```
//界面
void Interface();
//注册
void Reg(int sockfd);
//登陆
void Entry(int sockfd);
//客户端向服务器发送数据
Messges ask_ser(int sockfd);
#endif
```

3.3.2 操作实现

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include "clientlogin.h"

Messges msg;

//界面
void Interface(){
    printf("==============================\n");
    printf("       1.注册\t2. 登陆\n");
    printf("==============================\n");
}
//注册
void Reg(int sockfd){
    char name[20];
    char passwd[20];
    strcpy(msg.msg,"#hello");//发送消息hello
    printf("输入用户名：\n");
    scanf("%s",name);
    while(getchar()!='\n'){}
    strcpy(msg.name,name);

    printf("输入密码：\n");
    scanf("%s",passwd);
    while(getchar()!='\n'){}
    strcpy(msg.passwd,passwd);

    msg.cmd = 1;//定义为1，表示为注册消息
    write(sockfd,&msg,sizeof(msg));//将msg写入套接字，发送给服务器
    read(sockfd,&msg,sizeof(msg));//读取服务器的回复

    printf("msg.cmd = %d\n",msg.cmd);//打印服务回复的信息类型

    if(msg.cmd == 101){//回复码为101注册成功
        printf("注册成功！\n请稍后.......\n");
        sleep(3);
    }else if(msg.cmd == -1){
```

```
        printf("注册失败\n请稍后.....\n");
        sleep(3);
    }else if(msg.cmd == -3){
        printf("用户名已存在! \n请稍后");
        sleep(3);
    }
}
//登陆
void Entry(int sockfd){
    char name[20];
    char passwd[20];
    strcpy(msg.msg,"#hello");//发送消息hello
    printf("输入用户名: \n");
    scanf("%s",name);
    while(getchar()!='\n'){}
    strcpy(msg.name,name);

    printf("输入密码: \n");
    scanf("%s",passwd);
    while(getchar()!='\n'){}
    strcpy(msg.passwd,passwd);

    msg.cmd = 2;//2表示为登陆类型
    write(sockfd,&msg,sizeof(msg));//将msg写入套接字, 发送给服务器
    read(sockfd,&msg,sizeof(msg));//读取服务器的回复

    printf("msg.cmd = %d\n",msg.cmd);//打印服务回复的信息类型

    if(msg.cmd == 102){//回复码为102登陆成功
        printf("登陆成功! \n请稍后......\n");
        sleep(3);
    }else if(msg.cmd == -1){
        printf("登陆失败, 请检查账号或者密码! \n请稍后......\n");
        sleep(3);
    }else if(msg.cmd == -2){
        printf("用户不存在! \n请稍后......\n");
        sleep(3);
    }
}
//客户端向服务器发送注册/登录的选择
Messges ask_ser(int sockfd){
    char choice;
    Interface();
    printf("请选择1或者2\n");
    scanf("%c",&choice);
    switch (choice){
        case '1'://注册
            Reg(sockfd);
                break;
            case '2'://登陆
                Entry(sockfd);
                break;
    }
    system("clear");//清理屏幕输出
    return msg;
```

```
}
```

3.4 客户端功能实现

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <pthread.h>
#include "../include/clientlogin/clientlogin.h"
#include "../include/mysql_connection/mysql_connect.h"

#define PORT   7990
const char * HostIP = "192.168.203.137";
int sockcd;//客户端socket
char user_name[20];//用户名
char pass_wd[20];//用户密码
Messges msg;//引入自定义的消息类型，用于连接的
Messges msg_talk;//用于聊天信息

void* recv_pthread(void*p);
void online_num(MYSQL *conndb);

int main(){
    int ret;
    int choice;
    pthread_t tid;
    MYSQL *mysql_handle = NULL;//引入数据库
    //连接数据库
    mysql_handle = db_init(mysql_handle);
    //连接远程数据库
    mysql_handle =
db_connect(mysql_handle,"47.120.41.232","root","123456","chat");
    int create;//创建子线程专门用于接受消息
    sockcd = socket(AF_INET,SOCK_STREAM,0);
    typedef struct sockaddr SA;//引入结构体
    struct sockaddr_in addr;
    bzero(&addr,sizeof(addr)); //等同于memset(&ser_addr,0,sizeof(ser_addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(PORT);
    inet_pton(AF_INET, HostIP, &addr.sin_addr.s_addr);    //指定IP 字符串类型转换为
网络字节序 参3:传出参数
    //addr.sin_addr.s_addr = htonl(INADDR_ANY);
    ret = connect(sockcd,(SA*)&addr,sizeof(addr));
    if(ret == -1){
        perror("无法连接\n");
        exit(1);
    }
    printf("连接成功\n");
    //客户端向服务器发送注册/登录的选择

    msg = ask_ser(sockcd);
```

```
if(msg.cmd == -2){
    printf("\n");
    printf("%s用户还没有注册\n或者检查一下用户名是否输入错误吧\n",msg.name);
    printf("\n");
    exit(1);
}
if(msg.cmd == -1){
    printf("\n");
    printf("登陆失败！\n");
    printf("\n");
    exit(1);
}
if(msg.cmd == -3){
    printf("\n");
    printf("%s用户已注册\n",msg.name);
    printf("\n");
    exit(1);
}
//获取用户名
strcpy(user_name,msg.name);
if(msg.cmd == 101 || msg.cmd == 102){
    sprintf(msg.msg,"%s进入聊天室",msg.name);
    printf("%s进入聊天室\n",msg.name);
    msg.cmd = 0;
    write(sockcd,&msg,sizeof(msg));//写给服务端
}

printf("============================\n");
printf("       1.输入#hello，可选择查看状态\n");
printf("       2.输入#clear，可清除当前终端消息\n");
printf("       3.输入#exit，可退出登陆\n");
printf("       4.直接输入文字即可聊天（上三种特殊字符不可用于聊天）\n");
printf("============================\n");
//create = pthread_create(&tid,0,recv_pthread,0);
//if(create != 0){
//    printf("create pthread error\n");
//}
//pthread_detach(tid);
/*
printf("选择聊天类型\n1：私聊\t2：群聊\n");
int temp;
scanf("%d",&temp);
if(temp == 1){
    printf("当前在线的用户\n");
    online_num(mysql_handle);
    printf("请选择聊天的对象\n");
    char user_to_msg[20];
    scanf("%s",user_to_msg);
    if(!if_name_exist(mysql_handle,user_to_msg)){
        printf("%s用户不存在！\n",user_to_msg);
        exit(1);
    }
    strcpy(msg_talk.name,user_to_msg);
    printf("输入#exit退出登录\n");
```

```
        while(1){
            //输入发送的消息
            scanf("%s",msg_talk.msg);
            if(strcmp(msg_talk.msg ,"#exit")){
                sprintf(msg_talk.msg,"%s退出聊天室\n",user_name);
                strcpy(msg_talk.name,user_name);
                write(sockcd,&msg_talk,sizeof(msg_talk));
                close(sockcd);//关闭套接字
                exit(1);
            }
            msg_talk.cmd = 8;//私聊消息的状态码
            strcpy(msg.name,user_to_msg);
            write(sockcd,&msg_talk,sizeof(msg_talk));
            read(sockcd,&msg_talk,sizeof(msg_talk));
            if(msg_talk.cmd == 9){
            printf("%s:%s\n",msg_talk.name,msg_talk.msg);
            }
        }
    }*/
    create = pthread_create(&tid,0,recv_pthread,0);
    if(create != 0){
        printf("create pthread error\n");
    }
    pthread_detach(tid);
    while(1){
        //输入发送的消息
        scanf("%s",msg_talk.msg);
        msg_talk.cmd = 0;
        strcpy(msg_talk.name,user_name);
        //客户端退出
        if(strcmp(msg_talk.msg,"#exit") == 0){
            sprintf(msg_talk.msg,"%s退出聊天室\n",user_name);
            strcpy(msg_talk.name,user_name);
            write(sockcd,&msg_talk,sizeof(msg_talk));
            close(sockcd);//关闭套接字
            break;
        }else if(strcmp(msg_talk.msg,"#clear") == 0){
            system("clear");
        }else if(strcmp(msg_talk.msg,"#hello") == 0){
            printf("==============================\n");
            printf("       1.查看用户状态\n");
            printf("       2.设置在线状态\n");
            printf("       3.设置离线状态\n");
            printf("==============================\n");
            scanf("%d",&choice);
            switch(choice){
                case 1:
                    //msg_talk.cmd = 3;
                    //write(sockcd,&msg_talk,sizeof(msg_talk));
                    //read(sockcd,&msg_talk,sizeof(msg_talk));//从服务端读回信息
                    //msg_talk.cmd = 0;//重置状态码
                    /*printf("==============================\n");
                    for(int i =0 ;i<3;i++){
                        printf("%-10s",msg_talk.online[i]);
                    }
```

```
                    for(int j = 3;j<msg_talk.len;j++){

                    }
                    printf("\n");
                    printf("============================\n");*/
                    online_num(mysql_handle);
                    break;
                case 2:
                    msg_talk.cmd = 4;
                    strcpy(msg_talk.name,user_name);
                    write(sockcd,&msg_talk,sizeof(msg_talk));
                    msg_talk.cmd = 0;
                    printf("设置成功\n");
                    break;
                case 3:
                    msg_talk.cmd = 5;
                    strcpy(msg_talk.name,user_name);
                    write(sockcd,&msg_talk,sizeof(msg_talk));
                    msg_talk.cmd = 0;
                    printf("设置成功\n");
                    break;
            }
        }else{
            write(sockcd,&msg_talk,sizeof(msg_talk));//没有选择就直接发送信息
        }
    }
    return 0;
}

void* recv_pthread(void*p){
    Messges msg_talk2;
    while(1){
        if(msg_talk.cmd !=3){
            read(sockcd,&msg_talk2,sizeof(msg_talk2));
        }
        if(msg_talk.cmd == 0){
            printf("%s:%s\n",msg_talk2.name,msg_talk2.msg);
        }
    }
}//启动子线程处理服务端消息

void online_num(MYSQL *conndb){
    if(info_table(conndb,"select *from user_online")){
        //printf("查询成功!\n");
    }
}
```

3.5 服务端功能实现

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
```

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <pthread.h>
#include <errno.h>
#include <sys/wait.h>
#include "../include/mysql_connection/mysql_connect.h"
#include "../include/erro/erro.h"
#include "../include/serverlogin/serverlogin.h"

#define PORT   7990
const char * HostIP = "127.0.0.1";

int online_count = 0;//在线人数
int sockcd[100];//客户端套接字
MYSQL *mysql_handle = NULL;//引入数据库
Messges msg;//引入自定义的消息类型
void server_thread(void* p);//启动子线程处理消息

//创建一个数组用于保存线程号与用户名的关系
char user_my[100][20];

int main(){
    //创建套接字
    int sockfd;//服务端套接字
    int size = 100;//最大支持人数
    typedef struct sockaddr SA;//引入结构体
    struct sockaddr_in addr;
    pid_t pid;
    int ret;
    //-----------------
    //服务器建立
    sockfd = socket(AF_INET,SOCK_STREAM,0);
    if(sockfd == -1){
        info_erro("创建socket失败\n");
        exit(1);
    }
    bzero(&addr,sizeof(addr)); //等同于memset(&ser_addr,0,sizeof(ser_addr));
    int opt = 1;
    setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,(const void*)&opt,opt);//端口复用
    addr.sin_family = AF_INET;
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);
    //绑定IP以及端口
    ret = bind(sockfd,(SA*)&addr,sizeof(addr));
    if(ret == -1){
        perror("绑定失败\n");
        exit(1);
    }
    //监听客户端连接数量
    ret = listen(sockfd,100);
    if(ret == -1){
        perror("设置监听失败\n");
        exit(1);
    }
    //-----------------
```

```
    //连接数据库
    mysql_handle = db_init(mysql_handle);
    //连接远程数据库
    mysql_handle =
db_connect(mysql_handle,"47.120.41.232","root","123456","chat");
    printf("服务器设置成功! \n");
    //创建多线程
    while(1){
        //设置客户端
        struct sockaddr_in cliaddr;
        socklen_t len = sizeof(cliaddr);
        int fd = accept(sockfd,(SA*)&cliaddr,&len);
        if(fd == -1){
            perror("客户端连接失败\n");
            continue;
        }else{
            printf("连接成功\n");
        }
        for(int i=0;i<size;i++){
            if(sockcd[i]==0){
                //记录客户端的socket
                sockcd[i] = fd;
                printf("fd = %d\n",fd);
                //启动多线程，给客户端服务
                pthread_t tid;
                pthread_create(&tid,0,server_thread,&fd);
                pthread_detach(tid);//线程分离
                online_count++;//连接客户端数量
                break;
            }
        }
    }
    return 0;
}

void server_thread(void* p){
    int fd = *(int*)p;
    printf("pthread = %d\n",fd);
    while(1){
        int ret = read(fd,&msg,sizeof(msg));
        strcpy(user_my[fd],msg.name);//绑定线程号与用户名
        if(ret == -1){
            perror("read error");
            break;
        }else if(ret == 0){
            printf("客户端退出\n");
            //printf("发送的消息码是：%d\n",msg.cmd);
            //printf("发送的信息类型是：%s\n",msg.msg);
            drop_user_online(mysql_handle,msg.name);
            insert_user_online(mysql_handle,msg.name,"offline");
            pthread_exit(0);
            online_count--;
        }
        /*
        if(msg.cmd == 8){
```

```
            for(int i=0;i<100;i++){
                if(!strcmp(user_my[i],msg.name)){
                    msg.cmd = 9;
                    write(sockcd[i],&msg,sizeof(msg));
                    break;
                    //printf("%s:%s\n",msg.name,msg.msg);
                }
            }
        }
        */
        if(!strcmp(msg.msg,"#hello") && (msg.cmd > 0 && msg.cmd < 6)){
            switch(msg.cmd){
                case 1://注册
                    Reg(fd,msg,mysql_handle);
                    break;
                case 2://登陆
                    Entry(fd,msg,mysql_handle);
                    break;
                case 3://打印在线人数
                //info_table(mysql_handle,"select *from user_online");
                break;
                case 4://设置在线
                drop_user_online(mysql_handle,msg.name);//删除在线状态
                insert_user_online(mysql_handle,msg.name,"online");//添加离线状态
                break;
                case 5://设置离线
                drop_user_online(mysql_handle,msg.name);//删除在线状态
                insert_user_online(mysql_handle,msg.name,"offline");//添加离线状态
                break;
                case 8://私聊消息
                //查找需要聊天用户的tid;
                /*
                for(int i=0;i<100;i++){
                    printf("走的这个\n");
                    if(!strcmp(user_my[i],msg.name)){
                        msg.cmd = 9;
                        write(sockcd[i],&msg,sizeof(msg));
                        //printf("%s:%s\n",msg.name,msg.msg);
                    }
                }*/
                break;
            }
        }else {
            for(int i=0;i<online_count;i++){
                if(sockcd[i] != fd){//群发消息
                    write(sockcd[i],&msg,sizeof(msg));
                    printf("%s:%s\n",msg.name,msg.msg);
                }
            }
        }
    }
}

//gcc server.c ../include/mysql_connection/mysql_connect.c ../include/erro/erro.c
../include/serverlogin/serverlogin.c -o server `mysql_config --cflags --libs`
```

```
//gcc client.c ../include/clientlogin/clientlogin.c -o client
```

3.6 MySQL表结构

```
-- 创建两个表
-- 用户信息存放表
create table user(
    id int auto_increment primary key comment  'id号',
    name varchar(20) not null unique comment '用户名',
    passwd varchar(20) comment '密码'
);

-- 在线用户表
create table user_online(
    id int auto_increment primary key comment  'id号',
    name varchar(20) not null unique comment '用户名',
    state varchar(20) not null comment '状态'
);
```

3.7 程序状态码

```
cmd = 0;//默认码
cmd = 1;//注册消息码
cmd = 2;//登陆消息码
cmd = 3;//查询在线状态码
cmd = 4;//设置在线码
cmd = 5;//设置离线码

cmd = 8;//私聊消息的状态码
cmd = 9;//私聊消息的回复码
cmd = -1;//错误码

cmd = -2;//登陆用户名不存在错误回复码
cmd = -3;//注册用户名已存在错误回复码
cmd = 101;//注册成功回复码
cmd = 102;//登陆成功回复码
```

四、设计心得

1.了解了mysql在C语言中的工作流程

2.在mysql中直接调用C语言会存在一些问题，导致数据改写失败，不过知道了在C语言调用mysql可以通过特定的接口来避免这件事

3.加深了对socket的理解，清楚的知道了每个函数的参数以及相关的结构体

4.本次课程设计中使用CMake进行编译的，在探索的过程中了解了在Linux中动态链接库和静态链接库的使用

5.感谢魏老师的指导。