```
#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;

			case 9 :	//请求传输文件
				server_transfer_file(client_socket, msg);
```

```
                break;
            case 10:     //接受传输文件
                server_transfer_file_y(msg);
                break;
            case -10 : //拒绝传输文件
                server_transfer_file_n(msg);
                break;
            case 11:     //一切条件都已成立,直接开始传输
                server_start_transfer_file(msg);
        }
    }

    //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));
}


//传输文件
void server_transfer_file(int client_socket, Msg * msg)
{
        printf ("%s 请求发送%s 文件给 %s\n",msg->fromname,msg->signname,msg->localname);
        int i;
        for(i=0; i<NOW_MAX; i++)
        {
                if(user[i].socket != 0 && strcmp(user[i].name, msg->localname) == 0)
                {
                        write(user[i].socket, msg , sizeof(Msg));
                printf("发送给%s 信息进行判断是否接受\n",msg->localname);
                break;
            }
        }
        if (i == NOW_MAX)
        {
            msg->cmd == -9;       //表示传输文件失败
            write(client_socket, msg , sizeof(Msg));
            printf("发送文件失败,好友不在线或不存在\n");
        }
}

//接受文件
void server_transfer_file_y(Msg * msg)
{
        printf("%s  接受文件传输.\n",msg->fromname);

        int i;
        for(i=0; i<NOW_MAX; i++)
        {
            if(user[i].socket != 0 && strcmp(user[i].name, msg->localname) == 0)
            {
                write(user[i].socket, msg , sizeof(Msg));
                break;
            }
        }
}

//拒绝文件
void server_transfer_file_n(Msg * msg)
{
        printf("%s  不愿意接受文件传输.\n",msg->fromname);

        int i;
        for(i=0; i<NOW_MAX; i++)
        {
            if(user[i].socket != 0 && strcmp(user[i].name, msg->localname) == 0)
            {
                write(user[i].socket, msg , sizeof(Msg));
                break;
            }
        }
}

//一切条件都已成立,直接开始传输
```

```
void server_start_transfer_file(Msg * msg)
{
	//printf("文件传输中\n");
	int i;
	for(i=0; i<NOW_MAX; i++)
	{
		if(user[i].socket != 0 && strcmp(user[i].name, msg->localname) == 0)
		{
			write(user[i].socket, msg , sizeof(Msg));	//写文件数据
			break;
		}
	}
	if(msg->num != 1024)
	{
		printf("文件传输完成\n");
	}
}


int main()
{
	Make_Server ();
	return 0;
}
```