项目二聊天软件详细设计文档

## 功能模块设计概述

1. 初始化模块：初始化套接字、套接字队列、数据缓存区、用户信息块列表以及客户端的MFC窗口等。
2. 建立连接模块：服务器端使用bind、listen、accept等函数设置并接收新连接；客户端使用隐式绑定，以及connect与服务器建立连接。
3. 数据发送模块：使用send函数发送数据，数据传输格式按照事先约定好的数据结构。
4. 数据接收模块：使用recv函数接收数据，判断接收数据类型，若无数据，则判断是否关闭连接；若有数据，则以一定方式显示。
5. 用户数据信息管理模块：每一个连接建立时，将用户的套接字、昵称进行保存，并在之后的过程中将发往该用户的信息存储在该用户信息块中的接收缓存区中，当轮到该用户套接字时，将信息发送给用户。

## 基本需求

1. 编写两对简单的一对一聊天程序，实现任一用户可以向任一在线用户发送聊天信息。
2. 实现的聊天模式为随时发随时收的模式。
3. 实现用户上下线的提示与状态的改变
4. 实现可以同一时刻与多个用户分别聊天

## 关键问题分析及解决思路

1. 任一用户可以向任一在线用户发送聊天信息，可以通过一个中心服务器做聊天信息的中转，以此，用户只需知晓服务器的固定IP地址与端口号，便可以与任一用户通信
2. 实现的聊天模式为随时发随时收的模式，可以采用select进行多路复用，同时将套接字设置为非阻塞状态，当有套接字的读事件或写事件触发时，程序再进入处理相应事件
3. 实现用户上线下线状态改变，可以在正常的聊天信息中加入标识符指示该信息为控制信息，实现对用户在线状态的通知。
4. 实现可以同一时刻与多个用户分别聊天，可以通过在发送数据的头部加入目标用户在服务器端的套接字，来指示该聊天信息的发送目的地，从而通过改变头部大小就可以同时向多个用户发送聊天信息。

## 各子模块方案设计

1. 套接字管理模块，在服务器端实现对多个套接字的管理，包括初始化、插入、删除，以及生成系统套接字队列。
2. 用户信息管理模块，在服务器端实现对多个套接字所对应的用户的相关信息的管理，包括用户的插入、删除，广播用户上下线状态信息，告知新用户在线用户信息，以及转发用户之间的聊天信息。

## 报文格式及定义

struct msg\_block{

union{

struct{ SOCKET ID; char command; char msg\_buf[MSG\_BUF\_SIZE]; }msg;

char char\_buf[CHAR\_BUF\_SIZE];

};

};

ID：用于标识信息的目的或源用户。取用户与服务器建立连接后，在服务器端套接字值的大小。

command：用于标识该信息类型，正常聊天信息，或是上下线通知信息。（可扩充）

msg\_buf：用于存储信息内容。上限为MSG\_BUF\_SIZE（宏定义为1024，可更改）

## 子模块

* 1. **子模块：套接字管理模块**
     1. 子模块各相关处理函数描述

1. 初始化套接字队列函数：void init\_list(socket\_list \*list);
2. 插入新套接字函数：int insert\_list(SOCKET s, socket\_list \*list);
3. 删除套接字函数：int delete\_list(SOCKET s, socket\_list \*list);
4. 用套接字队列生成系统套接字队列：void make\_fdlist(socket\_list \*list, fd\_set \*fd\_list);
   1. **子模块：用户信息管理模块**
      1. 子模块各相关处理函数描述
5. 插入新用户块函数：void insert\_user\_list(user\_block \*user, SOCKET sock, char \*p\_nickname);
6. 插入用户新消息函数：int insert\_user\_list\_msg(user\_block \*user, msg\_block \*msg);
7. 广播用户状态函数：void broadcast\_user\_state(int n, socket\_list \*sock\_list, user\_block \*user\_list, char command);
8. 广播消息函数：void broadcast\_msg(int n, socket\_list \*sock\_list, user\_block \*user\_list, msg\_block \*origin\_info);
9. 通知新用户所有在线用户函数：void getall\_user\_state(int n, socket\_list \*sock\_list, user\_block \*user\_list);
10. 删除用户函数：void delete\_user\_list(user\_block \*user);

## 存在的问题

1. 客户端多窗口还未实现。未实现图片、表情的发送。
2. 服务器端没能实现select停等。

## 结语

体会了多路复用的基本流程，并完整的写出服务器和有界面的客户端。

## 参考文献