项目二聊天软件设计方案

版本号 V1.0

## 任务需求描述

1. 编写两对简单的一对一聊天程序，分别为面向连接方式和无连接方式
2. 当开始通信后，一方从键盘获得用户字符串，发送到另一方，然后等待对方数据，显示，再循环这个过程
3. 而另一方则是先等待对方字符串，显示；然后从键盘获得用户字符串，发送，再循环这个过程。

## 需求分析

1. 实现一对一聊天，本方案采用用户机-服务器-用户机的模式，即通过服务器转发用户机之间的聊天信息。任一用户可以向任一在线用户发送聊天信息
2. 针对无连接的方式，我们采用用户机-用户机（局域网下）的模式，即用户机上线后广播，通知在线的用户，从而建立连接。
3. 聊天的模式从简单的一发一收，进而向随时发随时收的模式转变，实现过程中需要使用线程，将收发分为两个线程分别执行。

## 关键技术分析

1. 在实现随时发随时收的目标时，我们需要将套接字设置为非阻塞状态，并用到select进行多路复用，使得客户机和服务器能及时处理用户的数据，当延时小于一定程度后，用户将感觉不到后台进行着并行处理。
2. 为了使客户端用户界面更加友好，本方案采用了MFC进行窗口的编写，将上线下线选择、用户列表、输入框、输出框都在界面上很好的呈现。
3. 在服务器端，为了更好地管理客户端连接的套接字，本方案使用了一个结构体socket\_list用于存储主套接字和用户套接字，并进行有效的管理。
4. 在服务器与客户端，为了更好地管理在线用户，本方案定义了一个结构体user\_block用于存储每一个用户的关键信息。

## 数据结构设计

1. 套接字列表

#define MAX\_SOCK\_NUM 16

struct socket\_list{

SOCKET MainSock;//主套接字

int num;//从套接字数量

SOCKET sock\_array[ MAX\_SOCK\_NUM ];//从套接字数组

};

1. 数据模块

#define MSG\_BUF\_SIZE 1024

#define CHAR\_BUF\_SIZE sizeof(SOCKET)/sizeof(char)+1+MSG\_BUF\_SIZE

struct msg\_block{

union{

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

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

};

};

//使用联合体，在结构体msg中，ID取服务器端与对应用户的套接字值，

//command标识本信息的类型（普通聊天信息'M'、上线信息'U'或下线信息'D'）

//msg\_buf存储信息内容，若信息类型为'M'，则为聊天信息；若信息类型为'U'或'D'，则为上下线用户昵称

//联合体另一个结构为字符串数组char\_buf，用于在发送时，直接作为发送的缓存区。

1. 用户信息块

#define NICKNAME\_LEN 20

#define RECV\_MSG\_LEN 16

struct user\_block{

SOCKET ID;//取服务器端与该用户的套接字值作为ID

char nickname[ NICKNAME\_LEN ];//该用户昵称

int msg\_num;//发往该用户的数据信息块的数量

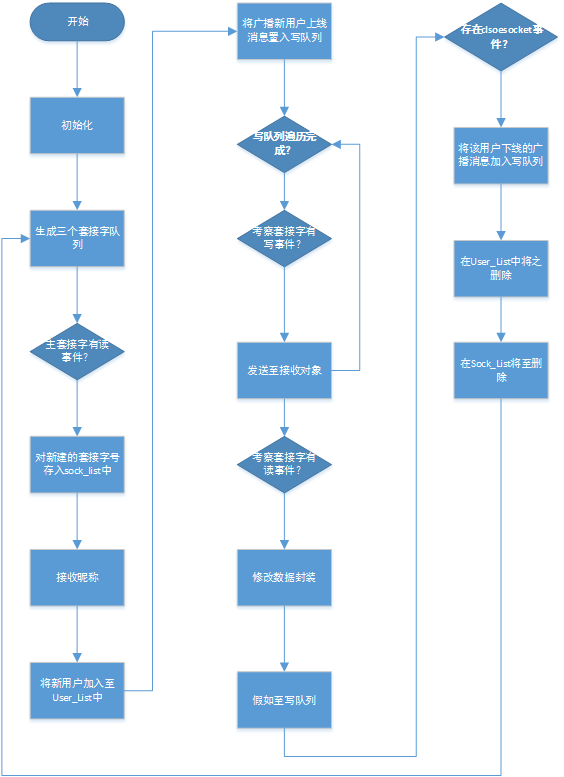
struct msg\_block \*p\_recv\_msg[ RECV\_MSG\_LEN ];//发往该用户的数据信息块的指针队列

};

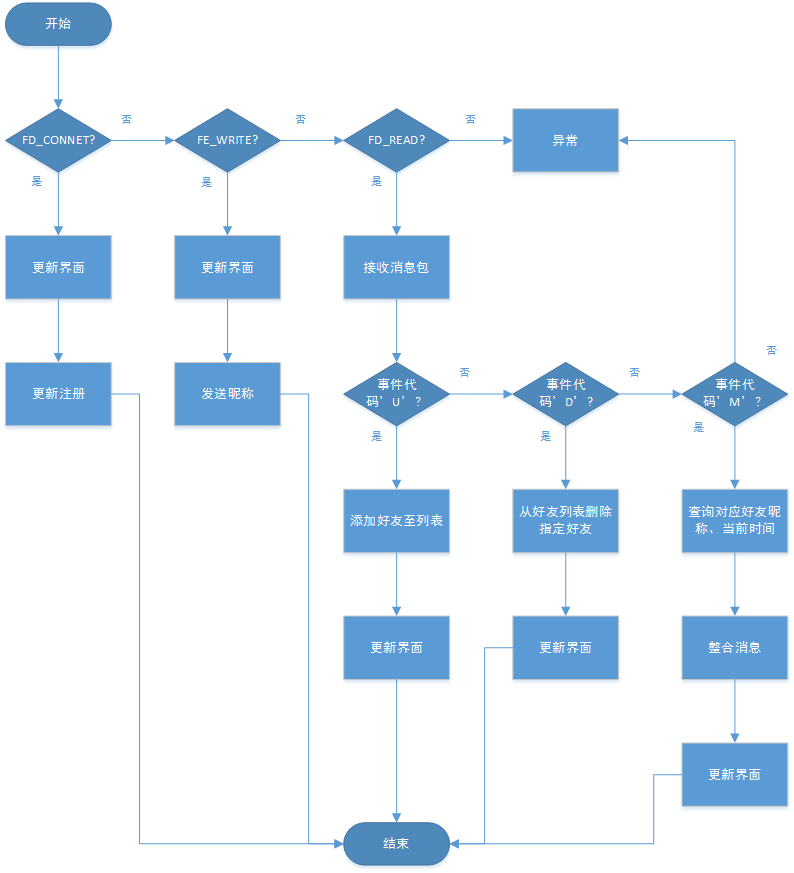
struct user\_block user\_list[ MAX\_SOCK\_NUM ];//用户信息块组成的用户队列

## 程序框架及流程设计

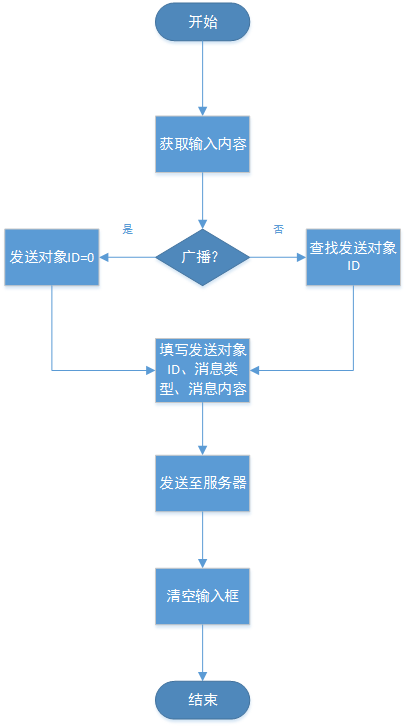
1. 面向连接的服务器：



1. 面向连接的客户机
   1. 消息处理流程图



1. 消息发送流程图



## 软件设计方案修订

暂无