之前一直不了解json的精髓，知道今天自己用结构体去组包去进行网络传输数据实现一个简单的聊天室

才明白json，两个字真香，json是一种轻量级的数据交换格式，它可以实现非常简单高效的把你想要定义

数据包的格式通过多个键值对的方式定义出来，并且可以通过成员方法转换成字符串的格式

服务端收到这个字符串的时候，再把这个字符串转换成一个json对象，通过自己定义的键值对拿到数据

就像剥洋葱一样一层层的拨开，很舒服。

用自己定义的结构体去组包很难受，一点都不方便，也许是自己定义的结构体不好，但是一想到json，我认为定义再好的结构体

也没有json方便。附赠我认为最简单方便快上手的json开源库地址：https://github.com/Bwar/CJsonObject

现在来说说自己写的这个udp聊天室和自己遇到了问题把

```
//包标志，定义的数据包类型，在分支结构里面处理不同的类型
enum ChatCommand
{
    C2S_LOGIN,      //上线              客户端的发包类型
    C2S_LOGOUT,     //下线
    C2S_GROUP,      //群聊包
    C2S_PRIVATE,    //私聊包

    S2C_LOGIN,      //上线              服务端的发包类型
    S2C_LOGOUT,     //下线
    S2C_GROUP,      //群聊包
    S2C_PRIVATE,    //私聊包
};
```

```
//用户的信息
#define NICKLEN 128
typedef struct tagUserInfo
{
```

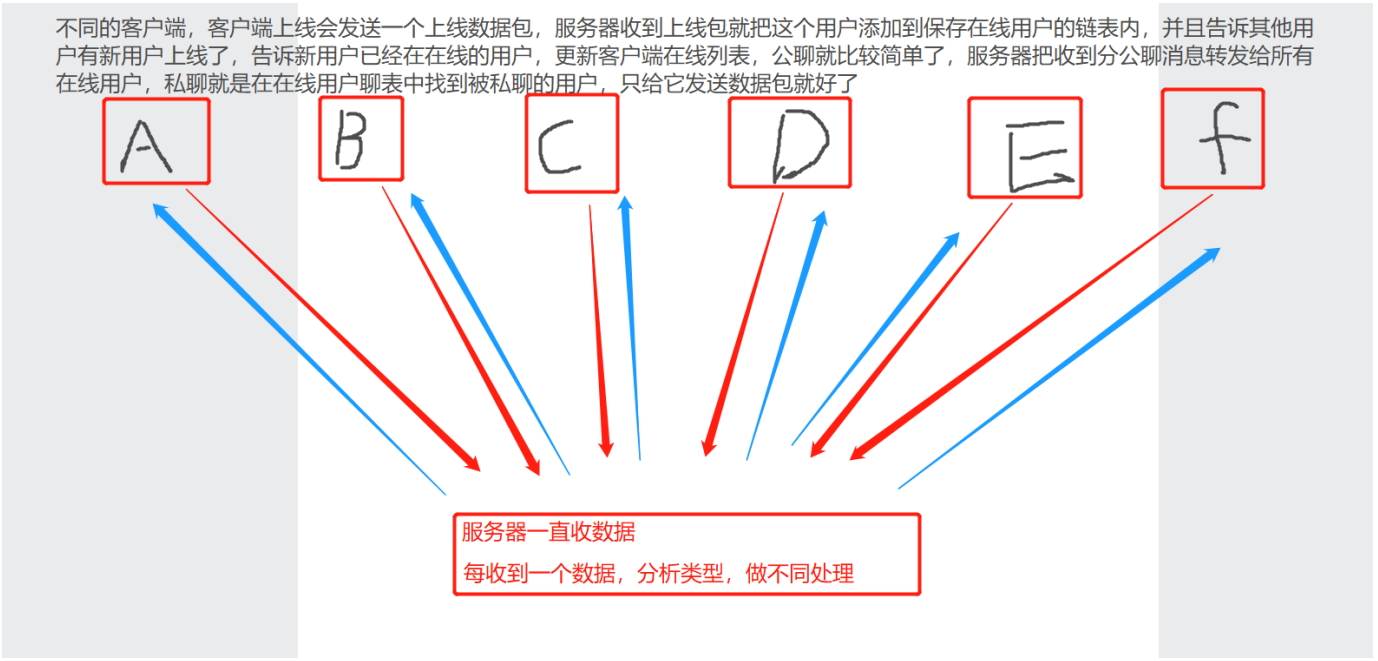
```
    WORD    _wPort;             //用户的端口号
    DWORD   _dwIp;              //用户的ip地址
    char    _szNick[NICKLEN];   //用户的名字
}USERINFO, * PUSERINFO;

#define MSGLEN 256
typedef struct tagPackage
{
    ChatCommand     _nCommand;                  //包的类型
    USERINFO        _ClientSend;                //发送信息的客户端
    USERINFO        _uiTo;                      //私聊中接受信息的客户端
    char            _szMsg[MSGLEN];             //发送的信息
}PACKAGE, * PPACKAGE;
//自己定义数据包成员，自认为成员名字和成员不是完美，但是能用
```

封装了一个发送数据包的函数

```
//          socket对象     接收端的IP和端口    数据包
bool SentTo(SOCKET sock, USERINFO& ui, PACKAGE& pkg)
{
    sockaddr_in si = { 0 };
    si.sin_family = AF_INET;
    si.sin_port = ui._wPort;
    si.sin_addr.S_un.S_addr = ui._dwIp;
    int nRet = sendto(sock, (char*)&pkg, sizeof(pkg), 0, (sockaddr*)&si, sizeof
    if (nRet == 0 || nRet == SOCKET_ERROR)
    {
        return false;
    }
    return true;
}
```

用一个简单的图来解释下这个聊天室



开线程收数据的时候，因为非UI线程不操作界面的原则，用自定义消息来处理收到的数据包，

先把收到的数据包依次存储在一个list中，因为出现了多个线程，所以用到了临界区，