

## 实验一：基于 socket 编程 TCP 聊天程序，通信内容采用

### DES 加密和解密过程

学号：2113683

姓名：柳致远

专业：密码科学技术

### 实验要求：

- 1、 利用 socket 编写一个 TCP 聊天程序。
- 2、 通信内容经过 DES 加密与解密

### 实验过程：

首先基于 socket 编写 TCP 聊天程序

#### 服务端：

全局变量声明如下

```
//=====全局变量区=====
const int BUFFER_SIZE = 1024; //缓冲区大小
int RECV_TIMEOUT = 10; //接收消息超时
int SEND_TIMEOUT = 10; //发送消息超时
const int WAIT_TIME = 10; //每个客户端等待事件的时间，单位毫秒
const int MAX_LINK_NUM = 10; //服务端最大链接数
SOCKET cliSock[MAX_LINK_NUM]; //客户端套接字 0号为服务端
SOCKADDR_IN cliAddr[MAX_LINK_NUM]; //客户端地址
WSAEVENT cliEvent[MAX_LINK_NUM]; //客户端事件 0号为服务端，它用于让程序的一部分等待来自另一部分的信号。
int total = 0; //当前已经链接的客户端服务数
```

为服务器分配 ip 地址和端口号

```
//3、将服务器地址打包在一个结构体里面
SOCKADDR_IN servAddr; //sockaddr_in 是internet环境下套接字的地址形式
servAddr.sin_family = AF_INET; //和服务器的socket一样，sin_family表示协议簇，一般用AF_INET表示TCP/IP协议。
servAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1"); //服务端地址设置为本地回环地址
servAddr.sin_port = htons(9999); //host to net short 端口号设置为12345

//4、绑定服务端的socket和打包好的地址
bind(servSock, (SOCKADDR*)&servAddr, sizeof(servAddr));
```

将消息转发给所有客户端：

```
//发送消息给所有客户端
while (1)
{
    char contentBuf[BUFFER_SIZE] = { 0 };
    char sendBuf[BUFFER_SIZE] = { 0 };
    cin.getline(contentBuf, sizeof(contentBuf));
    sprintf(sendBuf, "[柳哥bot:]%s", contentBuf);
    for (int j = 1; j <= total; j++)
    {
        send(cliSock[j], sendBuf, sizeof(sendBuf), 0);
    }
}
```

开启服务端线程并判断事件类型

若为接受事件则建立连接

```
WSANETWORKEVENTS networkEvents;
WSAEnumNetworkEvents(cliSock[i], cliEvent[i], &networkEvents); //查看是什么事件

//事件选择
if (networkEvents.lNetworkEvents & FD_ACCEPT) //若产生accept事件（此处与位掩码相与）
{
    if (networkEvents.iErrorCode[FD_ACCEPT_BIT] != 0)
    {
        cout << "连接时产生错误，错误代码" << networkEvents.iErrorCode[FD_ACCEPT_BIT] << endl;
        continue;
    }
}
```

若为客户端关闭事件，则断开连接并清除该客户端的资源

```
else if (networkEvents.lNetworkEvents & FD_CLOSE) //客户端被关闭，即断开连接
{
    //i表示已关闭的客户端下标
    total--;
    cout << " #" << i << "游客（IP: " << inet_ntoa(cliAddr[i].sin_addr) << "）退出了聊天室, 当前连接数: " << total << endl;
    //释放这个客户端的资源
    closesocket(cliSock[i]);
    WSACloseEvent(cliEvent[i]);

    //数组调整, 用顺序表删除元素
}
```

若为接受事件则将消息转发给其他客户端

```
}
else if (networkEvents.lNetworkEvents & FD_READ) //接收到消息
{
    char buffer[BUFFER_SIZE] = { 0 }; //字符缓冲区，用于接收和发送消息
    char buffer2[BUFFER_SIZE] = { 0 };

    for (int j = 1; j <= total; j++)
    {
        int nrecv = recv(cliSock[j], buffer, sizeof(buffer), 0); //nrecv是接收到的字节数
        if (nrecv > 0) //如果接收到的字符数大于0
        {
            sprintf(buffer2, "[%d] %s", j, buffer);
            //在服务端显示
            cout << buffer2 << endl;
            //在其他客户端显示（广播给其他客户端）
            for (int k = 1; k <= total; k++)
            {
                send(cliSock[k], buffer2, sizeof(buffer), 0);
            }
        }
    }
}
```

客户端：

首先打包服务器和客户端的 ip 地址以及端口号，并进行连接操作

```
//3. 打包地址
//客户端
SOCKADDR_IN cliAddr = { 0 };
cliAddr.sin_family = AF_INET;
cliAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); //IP地址
cliAddr.sin_port = htons(8888); //端口号

//服务端
SOCKADDR_IN servAddr = { 0 };
servAddr.sin_family = AF_INET; //和服务器的socket一样，sin_family表示协议簇，一般用AF_INET表示TCP/IP协议。
servAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1"); //服务端地址设置为本地回环地址
servAddr.sin_port = htons(9999); //host to net short 端口号设置为12345

bind(cliSock, (SOCKADDR*)&cliAddr, sizeof(cliAddr));
```

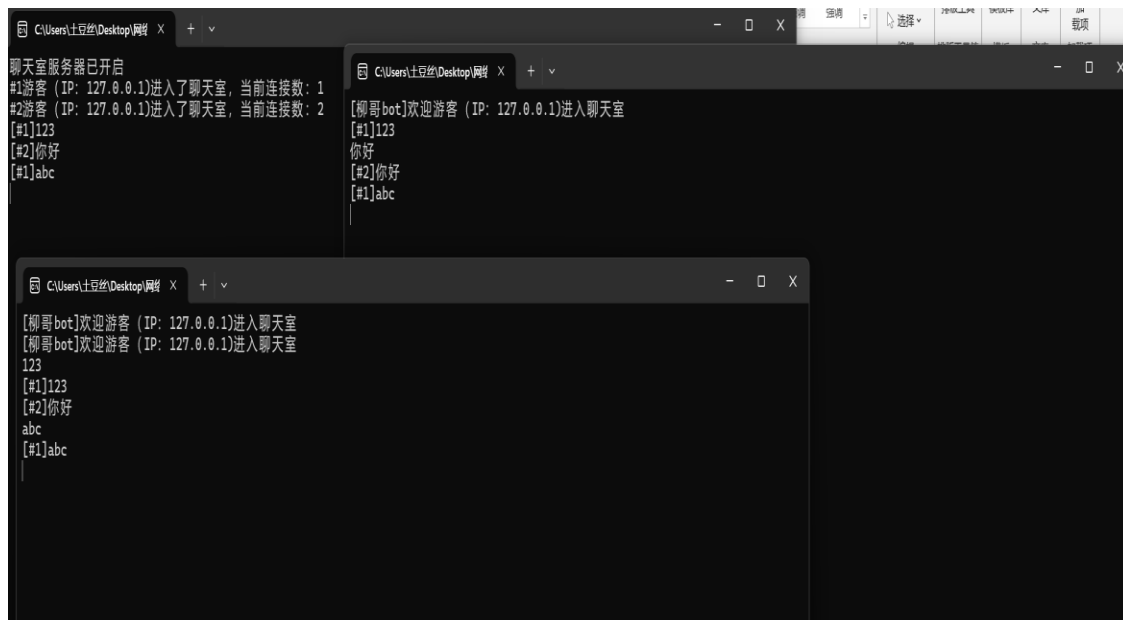
主线程用于发送要发送的消息

```
while (1)
{
    char buf[BUFFER_SIZE] = { 0 };
    cin.getline(buf, sizeof(buf));
    if (strcmp(buf, "quit") == 0)//若输入“quit”，则退出聊天室
    {
        break;
    }
    send(cliSock, buf, sizeof(buf), 0);
}
closesocket(cliSock);
WSACleanup();
return 0;
}
```

```
while (1)
{
    char buffer[BUFFER_SIZE] = { 0 };//字符缓冲区，用于接收和发送消息
    int nrecv = recv(cliSock, buffer, sizeof(buffer), 0);//nrecv是接收到的字节数
    if (nrecv > 0)//如果接收到的字符数大于0
    {
        des_decry(buffer);
        cout << MingWen << endl;
    }
    else if (nrecv < 0)//如果接收到的字符数小于0就说明断开连接
    {
        cout << "与服务器断开连接" << endl;
        break;
    }
}
return 0;
}
```

至此实现基于 socket 的 tcp 协议的聊天程序。

运行测试结果如下：



这里以开启两个客户端窗口为例。

测试成功后将聊天内容进行 DES 加解密操作

DES 加密的步骤为：

1、64 位明文初始置换

- 2、进行 16 轮置换与替代
- 3、进行初始置换的逆置换
- 4、得到 64 位密文

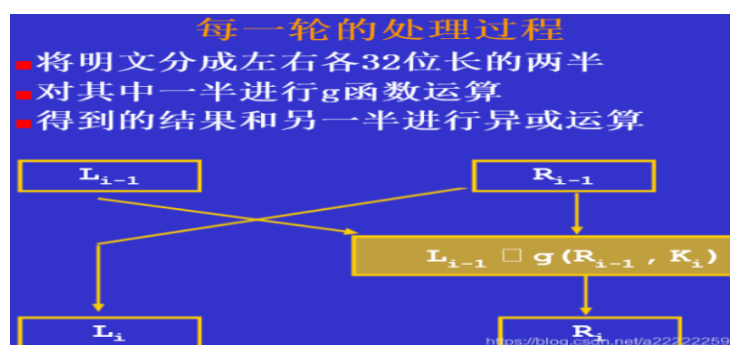
全局变量定义如下：

```
char MingWen[104]; //存放原始的明文
char target[8]; //将明文断成8个字符的一个分组
char InputKey[9]; //存放字符型的八位密钥
int text[64]; //存放一个分组转成二进制后的数据
int text_ip[64]; //存放第一次初始换位的结果
int L0[32], Li[32]; //将64位分成左右各32位进行迭代
int R0[32], Ri[32];
int RE0[48]; //存放右半部分经过E表扩展换位后的48位数据
int key[64]; //存放密钥的二进制形式
int keyPC1[56]; //存放密钥key经过PC1换位表后变成的56位二进制
int A[28]; //将keyPC1分成左右两部分，左部A，右部B，各28位，以便进行循环左
int B[28];
int keyAB[56]; //将循环左移后两部分的结果合并起来
int K[16][48]; //存放16次循环左移产生的子密钥
int RK[48]; //存放RE和K异或运算后的结果
int RKS[8]; //存放经过查找8个S表后得到的8个十进制结果
int SP[32]; //将RKS表中的十进制数化成二进制
int RKSP[32]; //存放SP表经过P盒换位后的结果
int text_end[64]; //存放经过左右32位换位后的结果
int text_out[14][64]; //存放初始化向量和所有经过DES的分组的二进制
char init[9] = { "HTmadeit" }; //设置初始化向量为“HTmadeit”
int CBC[64];
```

初始置换的置换表定义如下：

```
int IP[64] = { //初始换位表
    58, 50, 42, 34, 26, 18, 10, 2,
    60, 52, 44, 36, 28, 20, 12, 4,
    62, 54, 46, 38, 30, 22, 14, 6,
    64, 56, 48, 40, 32, 24, 16, 8,
    57, 49, 41, 33, 25, 17, 9, 1,
    59, 51, 43, 35, 27, 19, 11, 3,
    61, 53, 45, 37, 29, 21, 13, 5,
    63, 55, 47, 39, 31, 23, 15, 7
};
```

16 轮的置换替代过程为



换位表、移位表、s 盒均以数组的形成存储

```
int PC1[56] = { //PC1换位表 (64→56)
    57, 49, 41, 33, 25, 17, 9,
    1, 58, 50, 42, 34, 26, 18,
    10, 2, 59, 51, 43, 35, 27,
    19, 11, 3, 60, 52, 44, 36,
    63, 55, 47, 39, 31, 23, 15,
    7, 62, 54, 46, 38, 30, 22,
    14, 6, 61, 53, 45, 37, 29,
    21, 13, 5, 28, 20, 12, 4 };

int move[16] = { //循环移位表
    1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };

int PC2[48] = { //PC2换位表 (56→48)
    14, 17, 11, 24, 1, 5,
    3, 28, 15, 6, 21, 10,
    23, 19, 12, 4, 26, 8,
    16, 7, 27, 20, 13, 2,
    41, 52, 31, 37, 47, 55,
    30, 40, 51, 45, 33, 48
```

默认密钥为 12345678，并由此生成 16 轮的 key

```
void key_generate()
{
    strcpy(InputKey, "12345678");
    for (i = 0; i < 8; i++) //将密钥转化成64位二进制数放到一维数组key中
    {
        int a[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };
        m = InputKey[i];
        for (j = 0; m != 0; j++)
        {
            a[j] = m % 2;
            m = m / 2;
        }
        for (j = 0; j < 8; j++)
            key[(i * 8) + j] = a[7 - j];
    }
}
```

准备工作准备就绪后就可以开始编写 DES 加密以及解密函数，DES 加密解密过程存在对称性，加密过程解密过程将 16 轮密钥逆置即可实现解密操作。  
实现结果如下：

Client or Server? s Listening ... Server: got connection from 127.0.0.1, port 51424, socket 4 Send RSA public key successful! Begin chat ... Receive message from<127.0.0.1>: 123 Receive message from<127.0.0.1>: 159 Receive message from<127.0.0.1>: 1458 Receive message from<127.0.0.1>: dfc Receive message from<127.0.0.1>: ccba Receive message from<127.0.0.1>: hello	Client or Server? c Please input the server address: 127.0.0.1 Connect Success! Generate DES key successful! Send DES key successful! Begin chat ... 123 159 1458 dfc
---	--