****

**计算机网络实验报告**

Socket编程进阶



**学 院 国际工程学院**

**年 级 2018**

**专 业 计算机技术**

**学 号 2018229038**

**姓 名 李亚蓉**

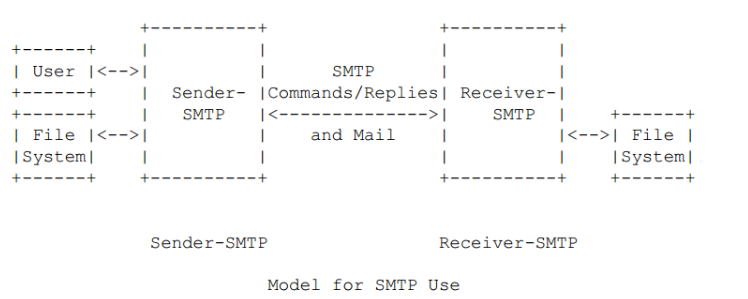
**2018年 12月16 日**

**Socket编程进阶**

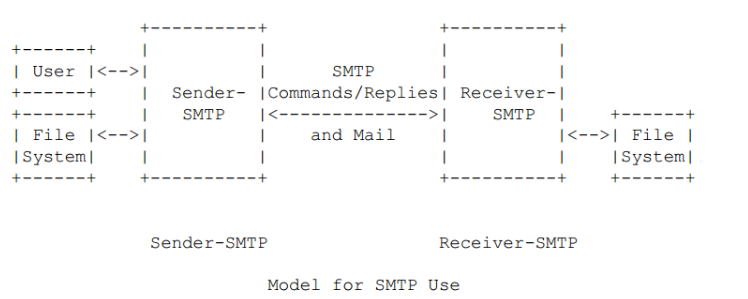
1. **实验目的**

使用Socket编程技术实现一个简单的smtp客户端软件，具有邮件发送能力。

1. **实验原理**
2. SMTP协议简介



SMTP称为简单邮件传输协议（Simple Mail Transfer Protocal），目标是向用户提供高效、可靠的邮件传输。它的一个重要特点是它能够在传送中接力传送邮件，即邮件可以通过不同网络上的主机接力式传送。通常它工作在两种情况下：一是邮件从客户机传输到服务器；二是从某一个服务器传输到另一个服务器。SMTP是一个请求/响应协议，它监听25号端口，用于接收用户的Mail请求，并与远端Mail服务器建立SMTP连接。

1. SMTP协议工作机制

SMTP通常有两种工作模式。发送SMTP和接收SMTP。具体工作方式为：发送SMTP在接收到用户的邮件请求后，判断此邮件是否为本地邮件，若是直接投送到用户的邮箱，否则向DNS查询远端邮件服务器的MX记录，并建立与远端接收SMTP之间的一个双向传送通道，此后SMTP命令由发送SMTP发出，由接收SMTP接收，而应答则反方向传送。一旦传送通道建立，SMTP发送者发送MAIL命令指明邮件发送者。如果SMTP接收者可以接收邮件则返回OK应答。SMTP发送者再发出RCPT命令确认邮件是否接收到。如果SMTP接收者接收，则返回OK应答；如果不能接收到，则发出拒绝接收应答（但不中止整个邮件操作），双方将如此反复多次。当接收者收到全部邮件后会接收到特别的序列，入伏哦接收者成功处理了邮件，则返回OK应答。

3、SMTP的连接和发送过程

（1）建立TCP连接

（2）客户端发送HELO命令以标识发件人自己的身份，然后客户端发送MAIL命令；

         服务器端正希望以OK作为响应，表明准备接收

（3）客户端发送RCPT命令，以标识该电子邮件的计划接收人，可以有多个RCPT行；

         服务器端则表示是否愿意为收件人接收邮件

（4）协商结束，发送邮件，用命令DATA发送

（5）以.表示结束输入内容一起发送出去

（6）结束此次发送，用QUIT命令退出

1. **实验过程**

一般情况下，一封邮件的发送的过程如下：

（1）发信人利用客户端编辑邮件，包括填写发信人邮箱、收信人的邮箱和邮件标题等等；

（2）客户端提取发信人的编辑的信息，生成一封符合邮件格式的标准邮件；

（3）客户端利用SMTP将邮件发送到发送端 的邮件服务器；

（4）发送端服务器用SMTP将邮件发送到接收端邮件服务器。

SMTP通信过程

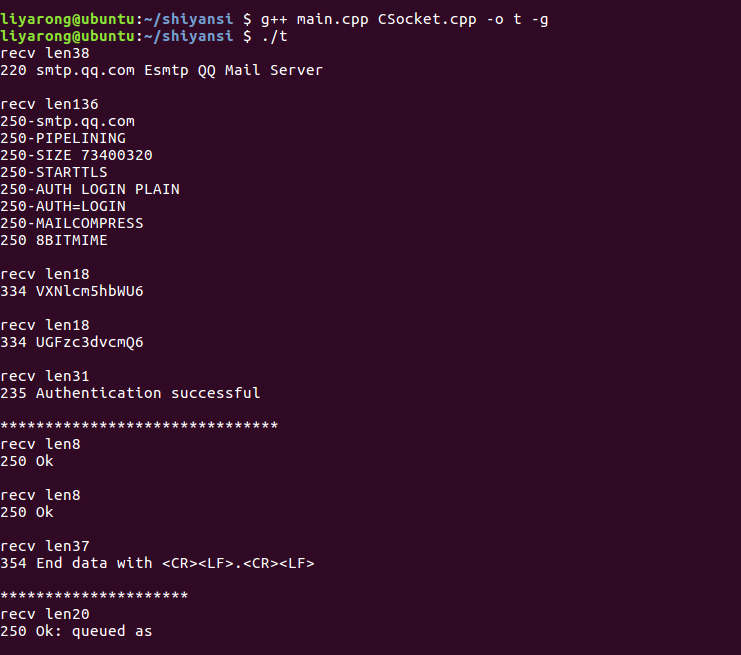
一个具体的SMTP通信的过程如下：

（1）发送端邮件服务器与接受端邮件服务器的固定端口建立TCP连接；

（2）客户端向服务器发送各种命令，来请求各种服务；

（3）服务器解析用户的命令，做出相应动作返回给客户端一个响应；

（4）（2）和（3）交替进行，知道所有的邮件都发送完或者意外中断。

**四、****实验结果**

图一 成功运行程序的显示

图二 成功发送邮件到收件人

1. **体会**

通过本次实验，更加清晰地了解了SMTP协议的连接和发送过程，虽然在实验的过程中遇到了一些小小的问题，但是在同学的帮助下成功地解决了问题，起初进行测试的邮箱是163邮箱，出现的问题是只允许自己跟自己发邮件，而不能发送到其他的邮箱上，经过修改，测试将163邮箱的服务IP改为QQ邮箱的服务IP，采用QQ邮箱即可，在绑定发送者的账号和密码时，应当将其利用base64进行加密，需要注意的是，打开QQ邮箱，设置->账户->开启POP3/IMAP/SMTP/Exchange/CardDAV,CalDAV服务，生成授权码，将授权码进行base64加密，结果为登录时的密码。希望下次再接再厉！

**六、源代码**

1. CSocket.h

#ifndef \_CSOCKET\_H\_

#define \_CSOCKET\_H\_

#include <sys/socket.h>

#include <arpa/inet.h>

#define BUFF\_LEN 256

class CSocket

{

public:

CSocket();

~CSocket();

bool initialize(const char\* ip, const int port);

bool sendSocketMessage(const char\* mesg);

bool reviceSocketMessage();

char\* getReviceMessage();

private:

int m\_sockfd;

char m\_reciveBuff[BUFF\_LEN];

struct sockaddr\_in m\_add;

};

#endif //\_CSOCKET\_H\_

1. CSocket.cpp

#include <unistd.h>

#include <string.h>

#include <memory.h>

#include <iostream>

#include "CSocket.h"

using namespace std;

CSocket::CSocket()

:m\_sockfd(-1)

{ }

CSocket::~CSocket()

{

if(-1 != m\_sockfd)

{

close(m\_sockfd);

}

}

bool CSocket::initialize(const char\* ip, const int port)

{

m\_sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if(-1 == m\_sockfd)

{

cerr<<\_\_FUNCTION\_\_<<endl;

return false;

}

m\_add.sin\_family = AF\_INET;

m\_add.sin\_port = htons(port);

int retPton = inet\_aton(ip, &m\_add.sin\_addr);

if(-1 == retPton)

{

cerr<<\_\_FUNCTION\_\_<<endl;

}

int retConn = connect(m\_sockfd, (struct sockaddr\*)&m\_add, sizeof(m\_add));

if(-1 == retConn)

{

cerr<<\_\_FUNCTION\_\_<<"connect error"<<endl;

}

}

bool CSocket::sendSocketMessage(const char\* mesg)

{

int ret = send(m\_sockfd, mesg, strlen(mesg), 0);

if(-1 == ret)

{

cerr<<\_\_FUNCTION\_\_<<"send message error"<<endl;

return false;

}

return true;

}

bool CSocket::reviceSocketMessage()

{

memset(m\_reciveBuff, 0x00, BUFF\_LEN);

int ret = recv(m\_sockfd, m\_reciveBuff, BUFF\_LEN, 0);

if(-1 == ret)

{

cerr<<\_\_FUNCTION\_\_<<"recv error"<<endl;

return false;

}

cout<<"recv len"<<ret<<endl;

return true;

}

char\* CSocket::getReviceMessage()

{

return m\_reciveBuff;

}

1. Main.cpp

#include "CSocket.h"

#include <iostream>

using namespace std;

int main()

{

CSocket sock;

sock.initialize("157.255.174.111", 25);

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

sock.sendSocketMessage("EHLO xxxxx\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

sock.sendSocketMessage("auth login\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

sock.sendSocketMessage("MTcyNzM2Nzc1");

sock.sendSocketMessage("\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

sock.sendSocketMessage("cHdpeXNkc2tsbmlmY2JiaA==");

sock.sendSocketMessage("\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

//发送方

sock.sendSocketMessage("mail from: <");

sock.sendSocketMessage("172736775@qq.com");

sock.sendSocketMessage(">");

sock.sendSocketMessage("\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

//接收方

sock.sendSocketMessage("rcpt to: <");

sock.sendSocketMessage("1542908241@qq.com");

sock.sendSocketMessage(">");

sock.sendSocketMessage("\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

sock.sendSocketMessage("data\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

//发送的邮件内容

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

sock.sendSocketMessage("subject:节日祝福\r\n\r\n");

sock.sendSocketMessage("新年快乐");

sock.sendSocketMessage("\r\n.\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

sock.sendSocketMessage("quit\r\n");

sock.reviceSocketMessage();

cout<<sock.getReviceMessage()<<endl;

}