**武汉大学计算机学院**

**本科生实验报告**

**使用Java/C++实现**

**POP客户端编程**

专 业 名 称 ：软件工程

课 程 名 称 ：网络及分布式计算实践

指 导 教 师 ：胡继承 职称 教授

学 生 学 号 ：2016302580159

学 生 姓 名 ：陈敬琳

二○一九年六月

**郑 重 声 明**

本人呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

本人签名： 陈敬琳 日期： 2019.6.11

摘 要

使用Java/C++实现POP客户端编程实验的实验目的是更加深入理解POP客户端的实现原理，加深学习印象，并能熟练掌握。

实验设计主要遵循Java编码原则和相关思想。

实验内容主要包括：使用Java完成user、pass、retr、list等命令，实现POP客户端编程。

实验结论为顺利完成实验，并掌握相关原理知识。

**关键词：**POP；Java编程；客户端

目录

[1 实验目的和意义 4](#_Toc11194091)

[1.1 实验目的 4](#_Toc11194092)

[1.2 实验意义 4](#_Toc11194093)

[2 实验设计 4](#_Toc11194094)

[2.1 概述 4](#_Toc11194095)

[2.2 实验原理 5](#_Toc11194096)

[2.3 实验方案 6](#_Toc11194097)

[结论 8](#_Toc11194098)

[附录 10](#_Toc11194099)

# 实验目的和意义

## 实验目的

本实验主要目的是掌握POP3的常见命令，如user, pass, quit等。熟悉利用socket建立TCP连接的基本过程。了解SMTP协议的基本内容。了解POP3协议的3种状态及其之间的转换。了解IMAP协议。

## 实验意义

本实验的意义在于，在掌握理论知识的基础上，动手实践，自己实现POP3的常见命令，加深对知识的理解，同时能够熟练应用，将理论和实践相结合。

# 实验设计

## 概述

实验环境：Intellij IDEA

参考：《计算机网络》POP3协议部分

本次实验使用Intellij IDEA为实验环境，参考《计算机网络》POP3协议部分，根据POP3协议的三种状态之间的转化，来达到实验目的，即使用Java实现POP客户端编程。

## 实验原理

POP3，全名为“Post Office Protocol - Version 3”，即“邮局协议版本3”。是TCP/IP协议族中的一员，由RFC1939 定义。本协议主要用于支持使用客户端远程管理在服务器上的电子邮件。提供了SSL加密的POP3协议被称为POP3S。

POP 协议支持“离线”邮件处理。其具体过程是：邮件发送到服务器上，电子邮件客户端调用邮件客户机程序以连接服务器，并下载所有未阅读的电子邮件。这种离线访问模式是一种存储转发服务，将邮件从邮件服务器端送到个人终端机器上，一般是PC机或 MAC。一旦邮件发送到 PC 机或MAC上，邮件服务器上的邮件将会被删除。但目前的POP3邮件服务器大都可以“只下载邮件，服务器端并不删除”，也就是改进的POP3协议。

POP3协议有三种状态，认正状态，处理状态，和更新状态,状态之间的转化如下：

客户机与服务器建立连接时，它的状态为认证状态；一旦客户机提供了自己身份并被成功地确认，即由认可状态转入处理状态； 在完成相应的操作后客户机发出QUIT命令（具体说明见后续内容），则进入更新状态，更新之后又重返认可状态；当然在认可状态下执行QUIT命令，可释放连接。具体如下图所示：

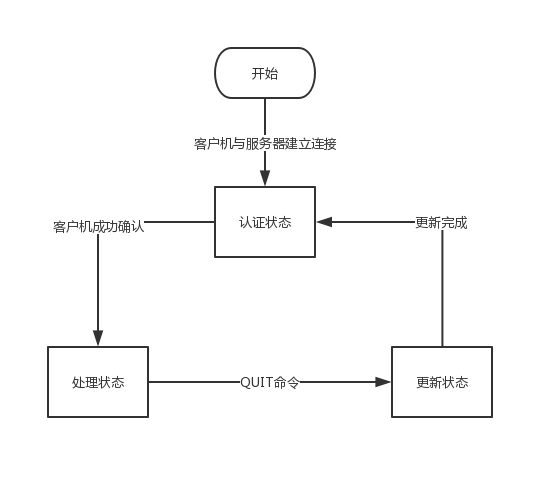
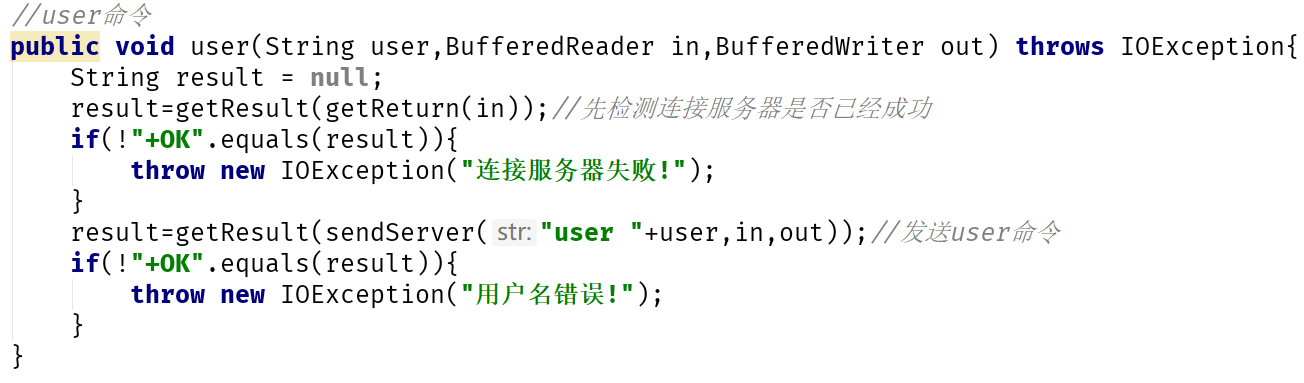


图2.1 POP3三种状态转化示意图

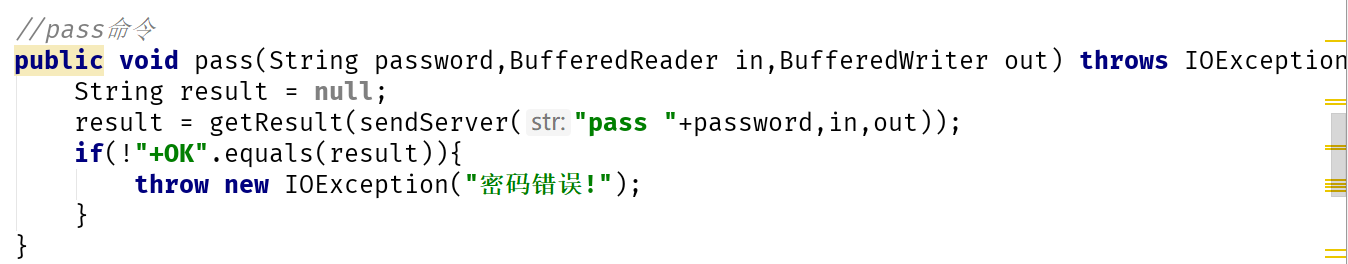
## 实验方案

为达到本次实验目的，本次实验如下九个步骤来实现。

1. 首先用tlenet程序连接到163的pop3邮箱，telnet pop3.163.com 110。
2. 执行user命令指定用户名。



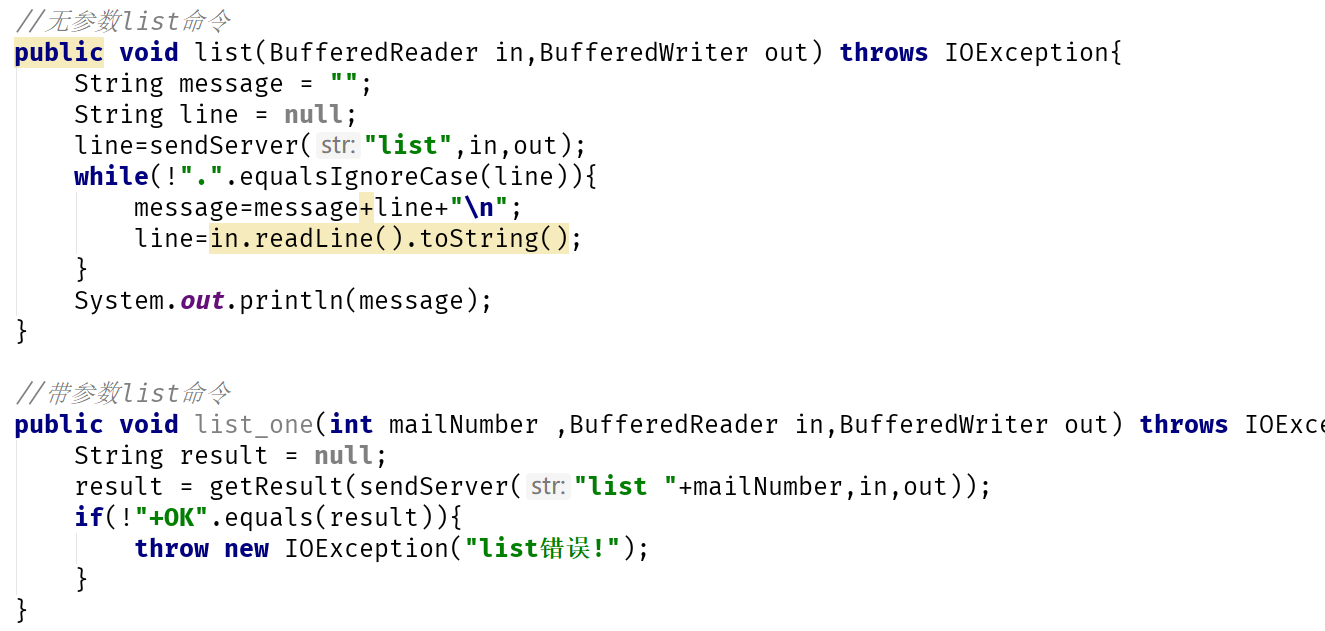
1. 执行pass命令输入密码。

验证成功后，提示邮箱中有一封邮件，占1822字节邮箱空间。

1. 执行stat命令统计邮箱中的信息，结果显示邮箱中有一封邮件，占1822字节的邮箱空间。



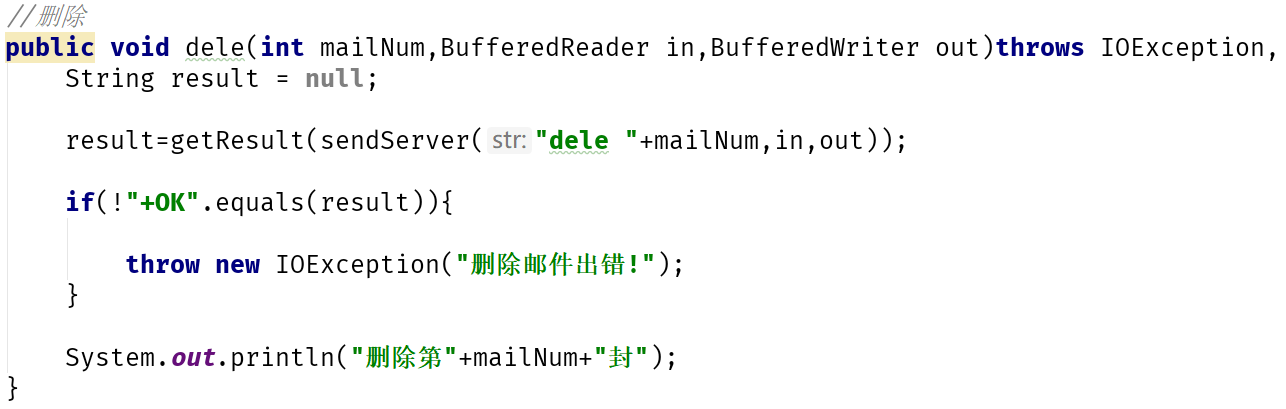
1. 执行list命令列出邮箱中的所有邮件，结果显示1 1822，1代表邮件编号，1822代表邮件的大小，如果有多封邮件，编号从1开始向上累加依次列出来。



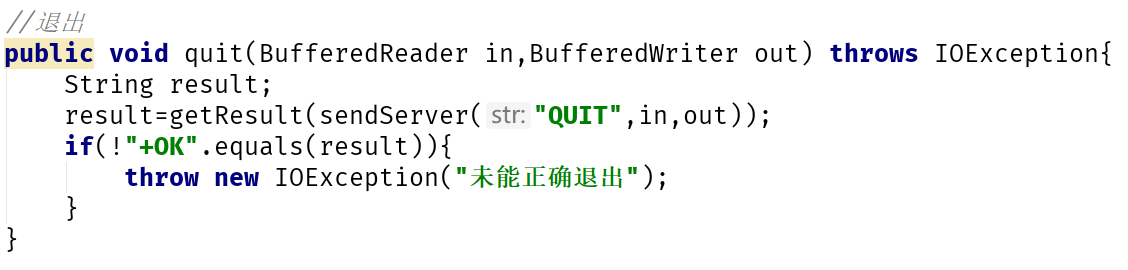
1. 执行retr 1命令查看第一封邮件的内容。



1. 执行dele 1命令将第一封邮件设置删除标志。

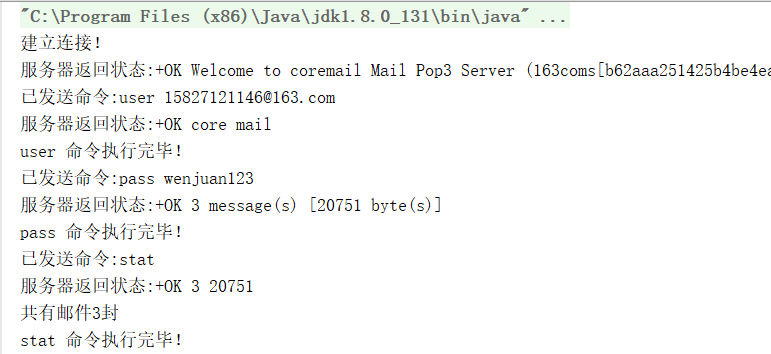


1. 执行rset命令重置所有邮件的删除标志。
2. 执行quit命令退出邮件接收程序，POP3服务器接收到客户端发送的quit命令后，将删除所有设置了删除标记的邮件，并断开与客户端的网络连接。并且Telnet程序自动结束运行，退回到Window命令行窗口状态。

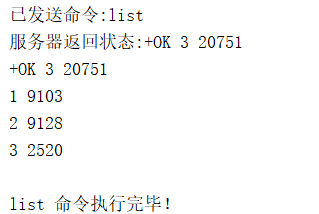


# 结论

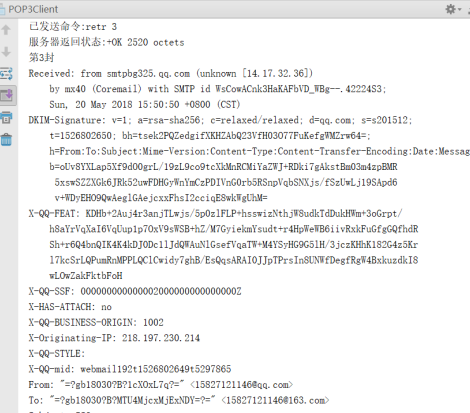
1. 通过如图所示的用户名和密码，成功连接邮箱，通过stat命令显示部分统计信息

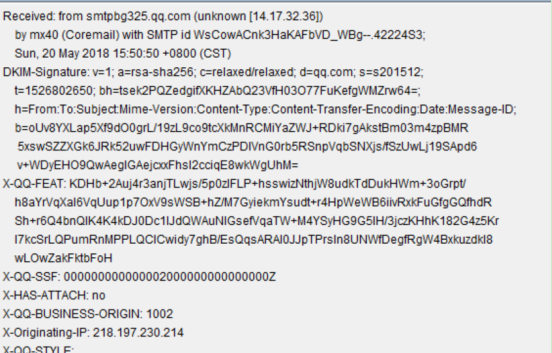


1. 通过list命令,显示每封邮件的大小信息



1. 通过retr命令显示第3封邮件的具体信息，并于Foxmail中打开该邮件，对比发现相同





1. 使用dele与quit命令删除邮件

# 附录

源代码部分：

POP3Client.java

import java.io.\*;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.StringTokenizer;

/\*\*

\* Created by Chen Jinglin on 2019/6/11.

\*/

public class POP3Client {

private Socket socket = null;

private boolean debug=true;

public static void main(String[] args) throws UnknownHostException, IOException {

String server="pop3.163.com";//POP3服务器地址

String user="15827121146@163.com";//用户名，填写自己的邮箱用户名

String password="cjl19980516.";//密码，填写自己的密码

POP3Client pop3Client=new POP3Client(server,110);

pop3Client.recieveMail(user,password);

}

/\*构造函数\*/

public POP3Client(String server,int port) throws UnknownHostException, IOException{

try{

socket=new Socket(server,port);//在新建socket的时候就已经与服务器建立了连接

}catch(Exception e){

e.printStackTrace();

}finally{

System.out.println("建立连接！");

}

}

//接收邮件程序

public boolean recieveMail(String user,String password){

try {

BufferedReader in=new BufferedReader(new InputStreamReader(socket.getInputStream()));

BufferedWriter out=new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

user(user,in,out);//输入用户名

System.out.println("user 命令执行完毕！");

pass(password,in,out);//输入密码

System.out.println("pass 命令执行完毕！");

stat(in,out);

System.out.println("stat 命令执行完毕！");

list(in,out);

System.out.println("list 命令执行完毕！");

retr(3,in,out);

System.out.println("retr 命令执行完毕！");

dele(3,in,out);

System.out.println("dele命令执行完毕");

quit(in,out);

System.out.println("quit 命令执行完毕！");

}catch(Exception e){

e.printStackTrace();

return false;

}

return true;

}

//得到服务器返回的一行命令

public String getReturn(BufferedReader in){

String line="";

try{

line=in.readLine();

if(debug){

System.out.println("服务器返回状态:"+line);

}

}catch(Exception e){

e.printStackTrace();

}

return line;

}

//从返回的命令中得到第一个字段,也就是服务器的返回状态码(+OK或者-ERR)

public String getResult(String line){

StringTokenizer st=new StringTokenizer(line," ");

return st.nextToken();

}

//发送命令

private String sendServer(String str,BufferedReader in,BufferedWriter out) throws IOException{

out.write(str);//发送命令

out.newLine();//发送空行

out.flush();//清空缓冲区

if(debug){

System.out.println("已发送命令:"+str);

}

return getReturn(in);

}

//user命令

public void user(String user,BufferedReader in,BufferedWriter out) throws IOException{

String result = null;

result=getResult(getReturn(in));//先检测连接服务器是否已经成功

if(!"+OK".equals(result)){

throw new IOException("连接服务器失败!");

}

result=getResult(sendServer("user "+user,in,out));//发送user命令

if(!"+OK".equals(result)){

throw new IOException("用户名错误!");

}

}

//pass命令

public void pass(String password,BufferedReader in,BufferedWriter out) throws IOException{

String result = null;

result = getResult(sendServer("pass "+password,in,out));

if(!"+OK".equals(result)){

throw new IOException("密码错误!");

}

}

//stat命令

public int stat(BufferedReader in,BufferedWriter out) throws IOException{

String result = null;

String line = null;

int mailNum = 0;

line=sendServer("stat",in,out);

StringTokenizer st=new StringTokenizer(line," ");

result=st.nextToken();

if(st.hasMoreTokens())

mailNum=Integer.parseInt(st.nextToken());

else{

mailNum=0;

}

if(!"+OK".equals(result)){

throw new IOException("查看邮箱状态出错!");

}

System.out.println("共有邮件"+mailNum+"封");

return mailNum;

}

//无参数list命令

public void list(BufferedReader in,BufferedWriter out) throws IOException{

String message = "";

String line = null;

line=sendServer("list",in,out);

while(!".".equalsIgnoreCase(line)){

message=message+line+"\n";

line=in.readLine().toString();

}

System.out.println(message);

}

//带参数list命令

public void list\_one(int mailNumber ,BufferedReader in,BufferedWriter out) throws IOException{

String result = null;

result = getResult(sendServer("list "+mailNumber,in,out));

if(!"+OK".equals(result)){

throw new IOException("list错误!");

}

}

//得到邮件详细信息

public String getMessagedetail(BufferedReader in) throws UnsupportedEncodingException{

String message = "";

String line = null;

try{

line=in.readLine().toString();

while(!".".equalsIgnoreCase(line)){

message=message+line+"\n";

line=in.readLine().toString();

}

}catch(Exception e){

e.printStackTrace();

}

return message;

}

//retr命令

public void retr(int mailNum,BufferedReader in,BufferedWriter out) throws IOException, InterruptedException{

String result = null;

result=getResult(sendServer("retr "+mailNum,in,out));

if(!"+OK".equals(result)){

throw new IOException("接收邮件出错!");

}

System.out.println("第"+mailNum+"封");

System.out.println(getMessagedetail(in));

Thread.sleep(3000);

}

//删除

public void dele(int mailNum,BufferedReader in,BufferedWriter out)throws IOException, InterruptedException{

String result = null;

result=getResult(sendServer("dele "+mailNum,in,out));

if(!"+OK".equals(result)){

throw new IOException("删除邮件出错!");

}

System.out.println("删除第"+mailNum+"封");

}

//退出

public void quit(BufferedReader in,BufferedWriter out) throws IOException{

String result;

result=getResult(sendServer("QUIT",in,out));

if(!"+OK".equals(result)){

throw new IOException("未能正确退出");

}

}

}