МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

Отчет   
по лабораторной работе   
на тему: « Взаимодействие прикладных программ с помощью

протокола электронной почты POP3»  
по дисциплине «Компьютерные сети»

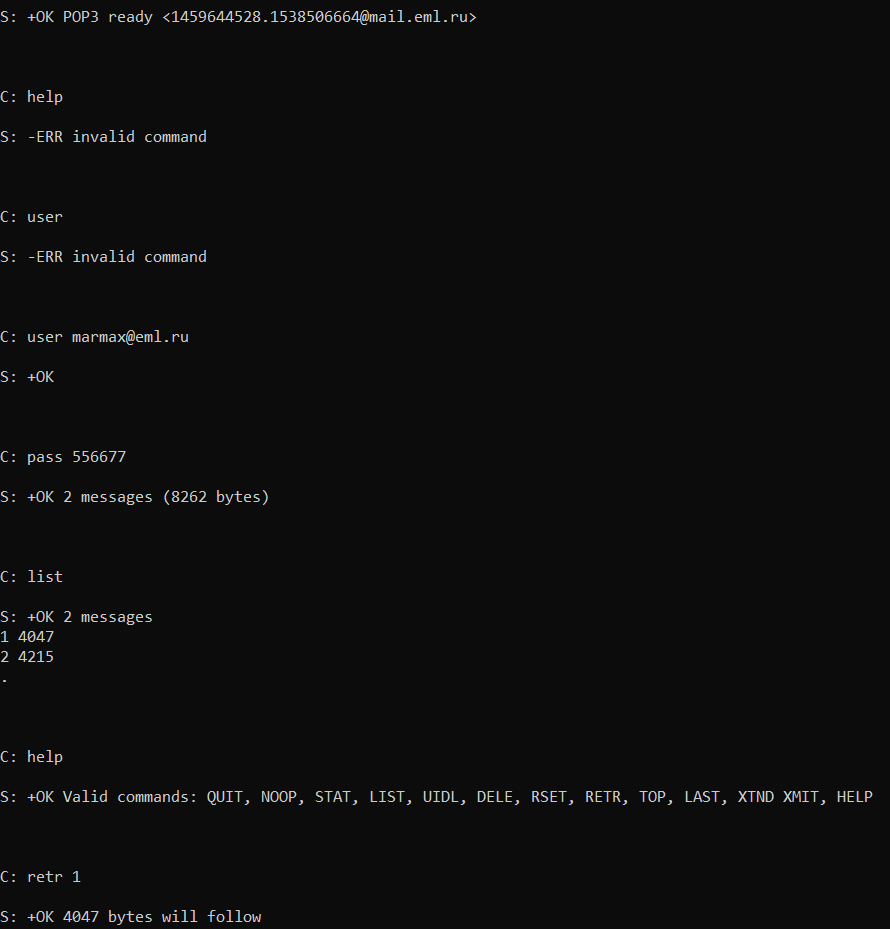
Выполнили: Марочкин М.А. Шифр: 170584   
 Шорин В.Д. Шифр: 171406  
ИПАИТ  
Направление: 09.03.04 «Программная инженерия»  
Группа: 71-ПГ  
Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Отметка о зачете:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

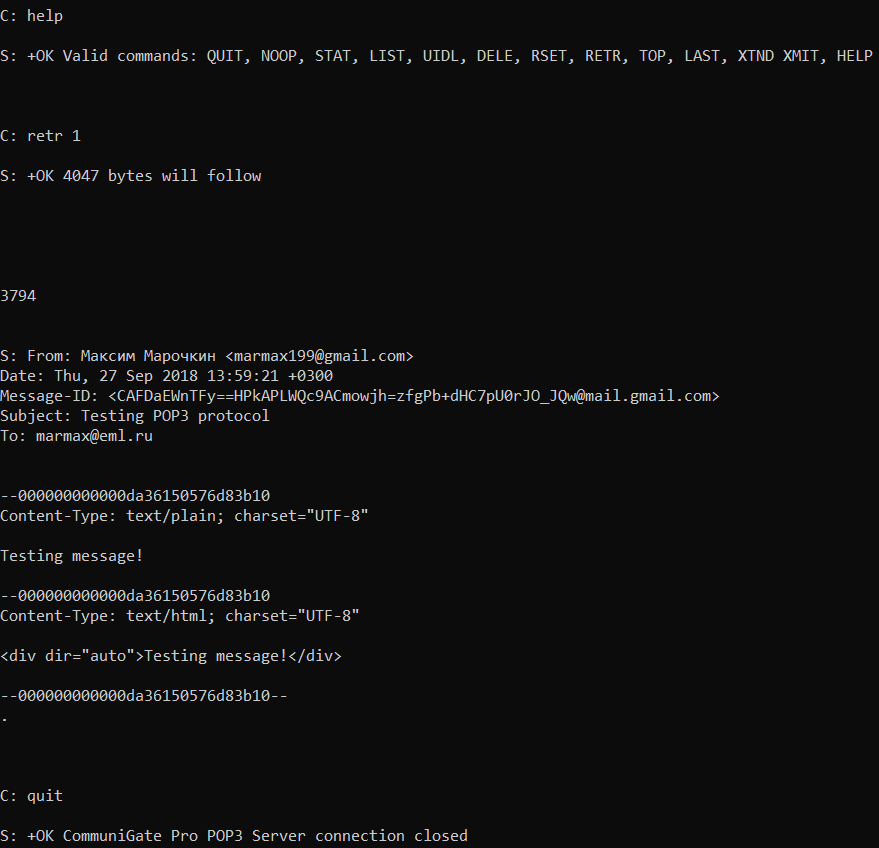
Дата «\_\_\_\_» \_\_\_\_\_\_ 2018 г.

Орел, 2018 г.

**Цель работы:**

Изучение принципов организации взаимодействия прикладных программ с помощью протокола электронной почты POP3 и приобретение практических навыков создания клиентских почтовых приложений, использующих протокол POP3.

**Выполнение работы:**

****

**Контрольные вопросы:**

1. **Охарактеризуйте модель протокола POP3.**

Post Office Protocol (POP) – протокол доставки почты пользователю из почтового ящика почтового сервера РОР. Многие концепции, принципы и понятия протокола POP выглядят и функционируют подобно SMTP. Команды POP практически идентичны командам SMTP, отличаясь в некоторых деталях. Сервер POP находится между агентом пользователя и почтовыми ящиками.

1. **Назовите и поясните назначение стадий РОР3-сессии.**

В протоколе РОР3 оговорены три стадии процесса получения почты: авторизация, транзакция и обновление. После того как сервер и клиент РОР3 установили соединение, начинается стадия авторизации. На стадии авторизации клиент идентифицирует себя для сервера. Если авторизация прошла успешно, сервер открывает почтовый ящик клиента и начинается стадия транзакции. В ней клиент либо запрашивает у сервера информацию (например, список почтовых сообщений), либо просит его совершить определенное действие (например, выдать почтовое сообщение). На стадии обновления сеанс связи заканчивается.

1. **Опишите формат почтового сообщения.**

Почтовое сообщение состоит из двух частей: заголовка и тела письма. Между ними расположена пустая строка. Каждое поле заголовка состоит в свою очередь из имени поля и значения, которые разделяются двоеточием. Полей заголовка может быть множество, но наиболее часто встречаются только некоторые из них. Поля To: и From: указывают на получателя и отправителя письма соответственно, поле Subject: используется для передачи темы письма. Reply To: применяется для указания обратного адреса, а CC: – копии сообщения. Received: – такую строку добавляет каждый почтовый сервис, через который прошло письмо. По нему легко отследить путь письма. Date: – означает дату создания письма. Используется специальный стандарт с указанием часового пояса отправителя относительно Гринвича.

Существует множество не стандартизованных полей заголовка. Они начинаются с символа X. Это могут быть такие поля, как X-Mailer: – почтовая программа отправителя; X-MSMail-Priority:, X-Priority: – приоритет (важность) письма и т.д. Разработчик также может создавать и использовать собственные поля, начинающиеся с X-.

1. **Назовите особенности и методы кодирования информации в почтовых сообщениях.**

Так как электронная почта изначально разрабатывалась как служба по пересылке обычного текста, это наложило ограничение на передаваемую информацию. Для передачи двоичной информации требуется специальное кодирование. Используется несколько методов кодирования: для текста с использованием только латинских букв, цифр и некоторых спецсимволов обычно используется Content-Transfer-Encoding: 7bit. Если сюда еще добавить русские буквы, то используется Content-Transfer-Encoding: 8bit. Для передачи HTML-кода применяют Content-Transfer-Encoding: quoted-printable. Чтобы отличить код от текста, перед ним ставится символ «=». Например, фраза «Привет!» будет выглядеть в закодированном виде как «=CF=F0=E8=E2=E5=F2!». Для двоичных файлов (архивы, изображения и т.д.) используют Content-Transfer-Encoding: base64, когда каждые 6 байт преобразуются в 8 печатных символов.

1. **По каким принципам формируются команды и коды ответов протокола РОР3?**

Команды POP3 состоят из ключевых слов, за некоторыми следует один или более аргументов. Все команды заканчиваются парой CRLF. Ключевые слова и аргументы состоят из ASCII символов. Ключевое слово и аргументы разделены одиночным пробелом. Ключевое слово состоит от 3-х до 4-х символов, а аргумент может быть длиной до 40 символов. Ответы в POP3 состоят из индикатора состояния и ключевого слова, за которым может следовать дополнительная информация. Ответ заканчивается парой CRLF. Существует только два индикатора состояния: «+OK» – положительный и «-ERR» – отрицательный.

1. **Какие операции выполняются на стадиях авторизации и обновления?**

*Стадия авторизации*

Как только будет установлено TCP соединение с POP3 сервером, он отправляет приглашение, заканчивающееся парой CRLF, например:

S: +OK POP3 server ready

Теперь POP3 сессия находится в режиме авторизации. Клиент должен идентифицировать себя на сервере, используя команды USER и PASS. Сначала необходимо отправить команду USER, после которой в качестве аргумента следует имя пользователя. Если сервер отвечает положительно, то необходимо отправить команду PASS, аргументом которой является пароль. Если после отправки команды USER или PASS сервер отвечает негативно, то можно попытаться авторизироваться снова или выйти из сессии с помощью команды QUIT.

После успешной авторизации сервер открывает и блокирует maildrop (почтовый ящик). В ответе на команду PASS сервер информирует, сколько сообщений находится в почтовом ящике, и передает их общий размер.

*Стадия обновления*

Когда клиент передает команду QUIT в режиме транзакции, то сессия переходит в режим обновления. В этом режиме сервер удаляет все сообщения, помеченные для удаления. После этого TCP соединение закрывается с соответствующим ответом сервера.

1. **Какие операции выполняются на стадии транзакции?**

После успешной идентификации пользователя на сервере POP3 сессия переходит в режим транзакции, где пользователь может передавать соответствующие команды (см. справочный раздел). После каждой из таких команд следует ответ сервера. В режиме транзакции доступны действия: получение статистической информации о корреспонденции, получение списка писем, чтение выбранного письма, чтение определенного количества строк текста выбранного письма, чтение идентификаторов всех или выбранного сообщения, пометка сообщения как удаленного, сброс отметок об удалении и др.

**Код:**

using System;using System.Collections.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using System.Net;using System.Net.Sockets;

namespace KC\_LABA\_POP3\_{

class Program {

static string Parser(string ans) {

int headerTail = ans.IndexOf("\r\n\r\n");

if (headerTail != -1) { Console.Write("\n\n" + headerTail + "\n\n"); }

int index = ans.IndexOf("From");

ans = ans.Remove(0, index);

index = ans.IndexOf(" ") + 1;

while (ans [index + 1] != ' ') { index++; }

string from = ans.Substring(ans.IndexOf(" ") + 1, index - ans.IndexOf(" ") + 1);

from = FromParser(from);

ans = ans.Remove(ans.IndexOf(" ") + 1, index - ans.IndexOf(" ") + 1);

ans = ans.Insert(ans.IndexOf("From") + 6, from + " ");

index = ans.IndexOf("To");

while (ans [index] != '\n') { index++; } index++;

ans = ans.Remove(index, ans.IndexOf("\r\n\r\n") - index);

return ans;

}

static string FromParser(string from) {

int index = from.IndexOf("B?") + 2;

from = from.Remove(0, index);

from = from.Substring(0, from.Length - 3);

var bytes = System.Convert.FromBase64String(from);

from = System.Text.Encoding.UTF8.GetString(bytes);

return from;

}

static void Main(string [] args) {

try {

byte [] bytes = new byte [40000];

byte [] msg = { };

int bytesRec;

int bytesSent;

string message = "";

string answer = "";

int port = 110;

IPHostEntry ipHost = Dns.GetHostEntry("mail.eml.ru");

IPAddress ipAddr = ipHost.AddressList [0];

IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, port);

Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

socket.Connect(ipEndPoint);

bytesRec = socket.Receive(bytes);

answer = Encoding.UTF8.GetString(bytes, 0, bytesRec);

Console.WriteLine("\nS: {0}\n\n", answer);

while (socket.Connected) {

Console.Write("C: ");

message = Console.ReadLine();

message += "\n";

msg = Encoding.UTF8.GetBytes(message);

bytesSent = socket.Send(msg);

bytesRec = socket.Receive(bytes);

answer = Encoding.UTF8.GetString(bytes, 0, bytesRec);

Console.WriteLine("\nS: {0}\n\n", answer);

if (answer.Contains("follow")) {

bytesRec = socket.Receive(bytes);

answer = Encoding.UTF8.GetString(bytes, 0, bytesRec);

answer = Parser(answer);

Console.WriteLine("\nS: {0}\n\n", answer);

}

answer = "";

}

}

catch (Exception ex) { Console.WriteLine(ex.Message); }

Console.Read();

}

}

}