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 SMTP »  
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:**

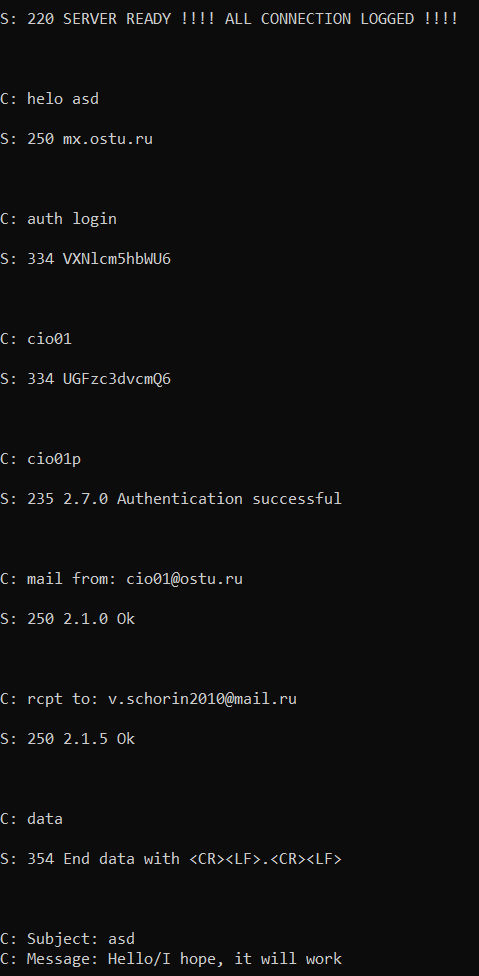
Изучение принципов организации взаимодействия прикладных программ с помощью протокола электронной почты SMTP и приобретение практических навыков создания клиентских почтовых приложений, использующих протокол SMTP.

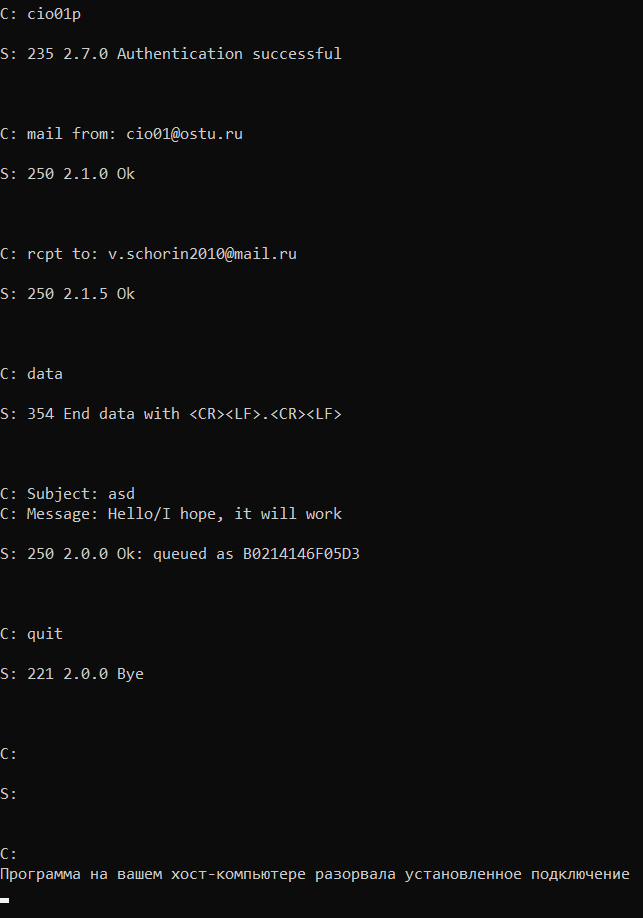
**Задание на лабораторную работу:**

Разработать приложение почтового клиента, позволяющее выполнять отправку текстовых сообщений по протоколу SMTP.

Для обмена командами и ответами с сервером используется транспортный протокол TCP/IP. Приложение должно поддерживать один из механизмов аутентификации: LOGIN или PLAIN. Приложение должно обеспечивать оценку кода ответа сервера на каждом этапе и осуществлять закрытие соединения, если код ответа отличается от требуемого. Сообщение должно включать поля From, To, Subject. Пользователь должен иметь возможность заполнить необходимые поля и дать команду на отправку сообщения. Процесс обмена командами и ответами должен происходить в автоматическом режиме и отображаться на экране в формате, приведенном в примере.

**Выполнение работы:**

****



Письмо пришло на почту:

C:\Users\Lenovo\Pictures\Screenshots\Снимок экрана (26).png

**Контрольные вопросы:**

**1. Назначение протокола SMTP.**

Для передачи сообщений по TCP-соединению большинство почтовых агентов пользуются протоколом SMTP (Simple Mail Transfer Protocol – простой протокол электронной почты). SMTP принят в качестве стандартного метода передачи электронной почты в сети Internet. Главной целью протокола SMTP является надежная и эффективная доставка электронных почтовых сообщений.

**2. Опишите модель работы протокола SMTP.**

Протокол SMTP базируется на следующей модели коммуникаций: в ответ на запрос пользователя почтовая программа-отправитель сообщения устанавливает двустороннюю связь с программой-приемником (почтовым сервером). Получателем может быть оконечный или промежуточный адресат. Если необходимо, почтовый сервер может установить соединение с другим сервером и передать сообщение дальше. SMTP-команды генерируются отправителем и посылаются получателю. На каждую команду должен быть получен отклик.

**3. Каковы особенности кодировки почтовых сообщений?**

Текст сообщения должен передаваться в виде 7-разрядных символов ASCII. Конец сообщения представляет собой строку, содержащую только символы точки «.» и перевода строки. Если по каким-то причинам такая строка непосредственно встречается внутри текста сообщения, то передающая сторона автоматически дублирует точку, чтобы принимающая сторона не приняла ее за конец сообщения. Большинство современных почтовых серверов поддерживают не только 7-битные, но и 8-битные кодировки, необходимые для кодирования текстов на национальных языках (в том числе русском), однако для совместимости со старыми почтовыми системами может использоваться специальная схема преобразования 8-битных кодов в 7-битные. Для того чтобы сообщение, посланное отправителем в одной кодировке, было понятно адресату, который применяет другую кодировку, почтовый сервер использует специальные таблицы смены кодировок. Чтобы сервер знал, какую кодировку использовать, информация об исходной кодировке включается почтовой системой отправителя в текст или в заголовок сообщения.

**4. Опишите типовую последовательность действий почтового клиента при отправке сообщения.**

Передача сообщения по протоколу SMTP происходит следующим образом: после установления соединения стороны обмениваются кодами аутентификации (с помощью команд HELO), затем одна из них посылает команду MAIL, при использовании которой необходимо указать получателя сообщения. Это делается при помощи команды RCPT. Команда RCPT имеет аргумент – имя получателя. На одну команду приходится только одно имя, поэтому, если получателей несколько, команда RCPT выдается несколько раз. Выдав команду RCPT, клиент ожидает получить ответ с кодом 250. После того как посланы все команды RCPT, клиент начинает передачу данных при помощи команды DATA. Клиент высылает команду DATA, сервер отвечает кодом 354. Этот код означает, что передача данных разрешена и должна заканчиваться комбинацией CRLF-точка-CRLF (новой строкой, содержащей только точку). После того как получен код 354, клиент может начать передачу данных. Сервер, в свою очередь, помещает принятые данные в очереди входящих сообщений. Сервер не высылает никаких ответов до тех пор, пока не получит комбинацию CRLF-точка-CRLF от клиента, означающую конец передачи данных. В ответ на полученную комбинацию CRLF-точка-CRLF сервер выдает код 250, который означает успешное окончание операции.

**5. Какие операции должны выполняться во время почтовой транзакции? Какими командами она открывается и закрывается?**

Когда SMTP-клиент устанавливает TCP-соединение с портом протокола 25, SMTP-сервер отвечает кодом 220. Это означает, что соединение успешно установлено.

После того как почтовые агенты компьютеров установили соединение и обменялись приветствиями, первой командой, согласно спецификации, должна быть команда HELO. SMTP-клиент передает HELO, указывая имя своего компьютера в качестве аргумента. В ответ на HELO приемник выдает код 250, сообщая передатчику о том, что команда принята и обработана.

После установления TCP-соединения и идентификации (при помощи HELO) SMTP-клиент приступает к почтовой транзакции. Для начала он выполняет одну из команд: MAIL, SEND, SOML или SAML.

Для того чтобы закончить почтовую транзакцию, клиент, по правилам SMTP, обязан послать команду QUIT. Сервер, в свою очередь, отвечает кодом 221. Этот код подтверждает клиенту, что соединение будет закрыто, после чего соединение действительно закрывается.

В любой момент во время транзакции клиент может использовать команды NOOP, HELP, EXPN и VRFY. В ответ на каждую команду сервер высылает клиенту определенную информацию.

*Команды протокола SMTP*

HELO (обязательна) Идентифицирует модуль-передатчик для модуля-приемника.

MAIL (обязательна) Начинает почтовую транзакцию, которая завершается передачей данных в один или несколько почтовых ящиков (mail).

RCPT (обязательна) Идентифицирует получателя почтового сообщения (recipient).

DATA Строки, следующие за этой командой, рассматриваются получателем как данные почтового сообщения. В случае SMTP, почтовое сообщение заканчивается комбинацией символов: CRLF-точка-CRLF.

RSET Прерывает текущую почтовую транзакцию (reset).

NOOP Требует от получателя не предпринимать никаких действий, а только выдать ответ ОК. Используется для тестирования. (No operation).

QUIT Требует выдать ответ ОК и закрыть текущее соединение.

VRFY Требует от приемника подтвердить, что ее аргумент является действительным именем пользователя.

SEND Начинает почтовую транзакцию, доставляющую данные на один или несколько терминалов (а не в почтовый ящик).

SOML Начинает транзакцию MAIL или SEND, доставляющую данные на один или несколько терминалов или в почтовые ящики.

SAML Начинает транзакцию MAIL и SEND, доставляющие данные на один или несколько терминалов и в почтовые ящики.

EXPN Команда SMTP-приемнику подтвердить, действительно ли аргумент является адресом почтовой рассылки, и если да, вернуть адрес получателя сообщения (expand).

HELP Команда SMTP-приемнику вернуть сообщение-справку о его командах.

*Коды ответа протокола SMTP*

211 – Ответ о состоянии системы или помощь.

214 – Сообщение-подсказка (помощь).

220 <имя\_домена> – Служба готова к работе.

221 <имя\_домена> – Служба закрывает канал связи.

250 – Запрошенное действие почтовой транзакции успешно завершилось.

251 – Данный адресат не является местным; сообщение будет передано по маршруту <forward-path>.

354 – Начинай передачу сообщения. Сообщение заканчивается комбинацией CRLF-точка-CRLF.

421 <имя\_домена> – Служба недоступна; соединение закрывается.

450 – Запрошенная команда почтовой транзакции не выполнена, так как почтовый ящик недоступен.

451 – Запрошенная команда не выполнена; произошла локальная ошибка при обработке сообщения.

452 – Запрошенная команда не выполнена; системе не хватило ресурсов.

500 – Синтаксическая ошибка в тексте команды; команда не опознана.

501 – Синтаксическая ошибка в аргументах или параметрах команды.

502 – Данная команда не реализована.

503 – Неверная последовательность команд.

504 – У данной команды не может быть аргументов.

550 – Запрошенная команда не выполнена, так как почтовый ящик недоступен.

551 – Данный адресат не является местным; попробуйте передать сообщение по маршруту <forward-path>.

552 – Запрошенная команда почтовой транзакции прервана; дисковое пространство, доступное системе, переполнилось.

553 – Запрошенная команда не выполнена; указано недопустимое имя почтового ящика.

554 – Транзакция не выполнена.

**Код:**

using System;using System.Collections.Generic;using System.Linq;using System.Text;using System.Net;

using System.Net.Sockets;using System.Threading.Tasks;

namespace KC\_LAB4{

class Program {

static void Main(string[] args) {

try {

byte [] bytes = new byte [1024]; byte [] msg = { };

int bytesRec; int bytesSent; string message; string answer = ""; int port = 25;

IPHostEntry ipHost = Dns.GetHostEntry("mail.oreluniver.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);

Console.WriteLine("\nS: {0}\n\n", Encoding.UTF8.GetString(bytes, 0, bytesRec));

while (socket.Connected) {

if (answer.Contains("354")) {

Console.Write("C: Subject: ");

message = "Subject: " + Console.ReadLine() + "\n";

Console.Write("C: Message: ");

message += Console.ReadLine(); message += "\r\n.\r\n";

}

else {

if (answer.Contains("334")) {

Console.Write("C: ");

message = Console.ReadLine();

msg = Encoding.UTF8.GetBytes(message);

message = Convert.ToBase64String(msg); message += "\n";

}

else {

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);

}

}

catch (Exception ex) {

Console.WriteLine(ex.Message);

}

Console.Read();

}

}

}