## Лабораторная работа №2 Протоколы SMTP и POP3

Лабораторная работа №2 выполняется после изучения материала, посвященного описанию принципов работы почтовых служб SMTP и POP3 [Компьютерные сети. / Э. Таненбаум, Коммпьютерные сети. / В. Олифер, Н. Олифер].

Цель работы:

написать GUI или консольное приложение для ОС Windows или Linux, реализующие работу протоколов SMTP или POP3 на стороне клиента.

Рекомендуемая литература:

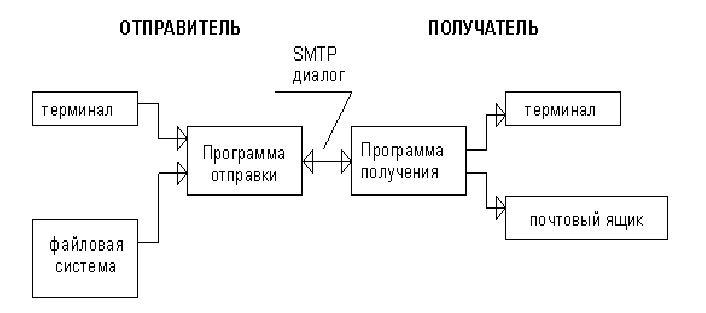
Описание протокола SMTP в спецификации RFC-788 (<https://www.ietf.org/rfc/rfc788.txt>), перевод: <https://rfc2.ru/5321.rfc> (ESMTP).

Описание протокола POP3 в спецификации RFC-1939 (<https://www.ietf.org/rfc/rfc1939.txt>), перевод: https://rfc2.ru/1939.rfc.

Протокол электронной почты SMTP Протокол SMTP (Simple Mail Transfer Protocol) был разработан для обмена почтовыми сообщениями в сети Internet. SMTP не зависит от транспортной среды и может использоваться для доставки почты в сетях с протоколами, отличными от TCP/IP и Х.25. Последнее обновление в RFC 5321 (2008) включает масштабируемое расширение — ESMTP (англ. Extended SMTP). В настоящее время под «протоколом SMTP» как правило подразумевают и его расширения.

Взаимодействие в рамках SMTP строится по принципу двусторонней связи, которая устанавливается между отправителем и получателем почтового сообщения. При этом отправитель инициирует соединение и посылает запросы на обслуживание, а получатель на эти запросы отвечает.

Фактически, отправитель выступает в роли клиента, а получатель - сервера. На рис. 2.1 приведена схема взаимодействия клиента и сервера по протоколу SMTP.

 Рис. 2.1 - Схема взаимодействия по протоколу SMTP

Канал связи устанавливается непосредственно между отправителем и получателем сообщения. При таком взаимодействии почта достигает абонента в течение нескольких секунд после отправки.

Обмен сообщениями и инструкциями в SMTP ведется в ASCII-кодах.

После установления соединения, как правило, используя 25 порт, клиент должен обязательно отправить на сервер команду

HELO <HOST>.

Эта команда используется для идентификации машины отправителя (HOST) на SMTP сервере.

Следующее командой должна идти команда MAIL, идентифицирующая отправителя:

MAIL <SP> FROM: <reverse-path> <CRLF>

Пример:

MAIL FROM: [cat@australia.mail.au](mailto:cat@australia.mail.au)

Эта команда указывает SMTP-серверу начать новую транзакцию по приёму почты. В качестве аргумента, она передаёт на сервер почтовый адрес отправителя письма.

Если адрес отправителя правильный и не содержит ошибок, то сервер вернёт ответ «250 OK».

Следующей командой идёт команда

RCPT:

RCPT <SP> TO: <forward-path> <CRLF>

Пример:

RCPT TO: [dog@switzerland.mail.sz](mailto:dog@switzerland.mail.sz)

Эта команда передаёт на сервер почтовый адрес получателя письма.

Если адрес получателя не содержит ошибок, то тогда SMTP сервер вернёт ответ «250 OK». Если в адресе получателя есть ошибка, то сервер вернёт сообщение с кодом 550. Данная команда может повторяться сколь угодно долго по числу получателей, однако современные почтовые сервера вводят ограничения на количество одновременных получателей.

Следующей командой идёт команда DATA <CRLF>

Если она принимается сервером, то он возвращает сообщение с кодом 354, приглашающее продолжить отправку сообщения. После этого, на сервер можно передавать текст почтового сообщения. Признаком окончания передачи почтового сообщения является символ точки «.» в начале новой строки. Если сообщение принято к доставке, то сервер вернёт уведомление с кодом 250, а иначе – сообщение об ошибке.

После принятия сервером сообщения к отправке, клиент должен отправить команду QUIT, которая сигнализирует серверу, что больше отправки писем не будет. После принятия от сервера подтверждения этой команды, следует закрыть соединение с сервером.

Пример SMTP диалога, между отправителем (SENDER) и сервером (RECEIVER):

SENDER: MAIL FROM: <Smith@Alpha>

RECEIVER: 250 OK

SENDER: RCPT TO: <Jones@Beta>

RECEIVER: 250 OK

SENDER: RCPT TO: <Green@Beta>

RECEIVER: 550 No such user here

SENDER: RCPT TO: <Brown@Beta>

RECEIVER: 250 OK

SENDER: DATA

RECEIVER: 354 Start mail input; end with <CRLF>.<CRLF>

SENDER: Blah blah blah...

SENDER: ...etc. etc. etc.

SENDER: <CRLF>.<CRLF>

RECEIVER:

250 OK

Отличие расширенного протокола ESMTP включает набор различных способов аутентификации, AUTH-LOGIN, CRAM-MD5, DIGEST-MD5.

Наиболее распространенный 'AUTH LOGIN' механизм выглядит так

S: 220 esmtp.example.com ESMTP

C: ehlo client.example.com

S: 250-esmtp.example.com

S: 250-PIPELINING

S: 250-8BITMIME

S: 250-SIZE 255555555

S: 250 AUTH LOGIN PLAIN CRAM-MD5

C: auth login

S: 334 VXNlcm5hbWU6

C: avlsdkfj

S: 334 UGFzc3dvcmQ6

C: lkajsdfvlj

S: 535 authentication failed

Как видно из примера, сначала посылается команда auth login, в ответ от сервера в кодировке base64 приходит сообщение Username:, после чего посылается логин в base64, затем от сервера приходит сообщение Password:, клиент отправляет свой пароль в base64.

Следует учесть, что большинство серверов в настоящее время поддерживают только защищенное SSL, TLS соединение. То есть, передаваемые команды должны шифроваться и передаваться по TCP протоколу. Кодировка base64 использует соответственно 64 символа, символы латинского алфавита (заглавные, прописные) и еще некоторые. Соответственно 3 символа байтового алфавита заменяются на 4 символа base64, для выравнивания по границе используется символ =.

### Формат электронного письма

Электронное письмо состоит из следующих частей:

Заголовков SMTP-протокола, полученных сервером. Эти заголовки могут включаться, а могут и не включаться в тело письма в дальнейшем, так что возможна ситуация, когда сервер обладает большей информацией о письме, чем содержится в самом письме. Так, например, поле RCPT TO указывает получателя письма, при этом в самом письме получатель может быть не указан. Эта информация передаётся за пределы сервера только в рамках протокола SMTP, и смена протокола при доставке почты (например, на узле-получателе в ходе внутренней маршрутизации) может приводить к потере этой информации. В большинстве случаев эта информация не доступна конечному получателю, который использует не SMTP протоколы (POP3, IMAP) для доступа к почтовому ящику. Для возможности контролировать работоспособность системы эта информация обычно сохраняется в журналах почтовых серверов некоторое время.

Самого письма (в терминологии протокола SMTP — 'DATA'), которое, в свою очередь, состоит из следующих частей, разделённых пустой строкой:

Заголовков письма, иногда называемых по аналогии с бумажной почтой конвертом. В заголовке указывается служебная информация и пометки почтовых серверов, через которые прошло письмо, пометки о приоритете, указание на адрес и имя отправителя и получателя письма, тема письма и другая информация. С термином «конверт» есть некоторая путаница, потому что в зависимости от ситуации «конвертом» называют либо заголовок письма, либо информацию, которой располагает SMTP-сервер после получения письма.

Тело письма. В теле письма находится, собственно, текст письма. Согласно стандарту, в теле письма могут находиться только символы ASCII. Поэтому при использовании национальных кодировок или различных форм представления информации (HTML, RTF, бинарные файлы) текст письма должен кодироваться по стандарту MIME и не может быть прочитан человеком без использования декодера или почтового клиента с таким декодером.

Заголовок SMTP

Заголовок SMTP содержит в себе следующую информацию:

имя отправляющего узла (не имя отправителя, а имя сервера или компьютера пользователя, который обратился к серверу) — параметр сообщения HELO/EHLO, обычно дополняющийся «объективной» информацией самим сервером (HELO может содержать произвольное имя, а IP отправителя подделать существенно сложнее), по IP-адресу осуществляется поиск PTR-записи в DNS, всё это вместе позволяет идентифицировать отправителя на сетевом уровне.

Поле MAIL FROM:, содержащее адрес отправителя. Адрес может быть произвольным (в том числе с несуществующих доменов, однако этот адрес может также проверяться при первичной проверке на спам).

Поле RCPT TO: — наиболее важное поле для доставки почты, содержит электронный адрес получателя. Большинство почтовых систем в случае возможности проверяет, существует ли пользователь и может отказаться принимать почту, если пользователь, указанный в RCPT TO не существует.

Заголовок письма

Заголовок отделяется от тела письма пустой строкой. Заголовок используется для журналирования прохождения письма и служебных пометок. В Microsoft Outlook этот заголовок называется «Заголовки Интернет». В заголовке обычно указываются: почтовые серверы, через которые прошло письмо (каждый почтовый сервер добавляет информацию о том, от кого он получил это письмо), информацию о том, похоже ли это письмо на спам, информацию о проверке антивирусами, уровень срочности письма (может меняться почтовыми серверами). Также в заголовке обычно пишется программа, с помощью которой было создано письмо. Чаще всего почтовые клиенты скрывают заголовки от пользователя при обычном использовании почтовой системой, но предоставляют возможность увидеть заголовки, если возникает потребность в более детальном анализе письма. В случае, если письмо из SMTP формата конвертируется в другой формат (например, в Microsoft Exchange 2007 письма конвертируются из SMTP-формата в MAPI), то заголовки сохраняются отдельно, для возможности диагностики.

Заголовки обычно добавляются снизу вверх (то есть каждый раз, когда к сообщению нужно добавить заголовок, он дописывается первой строкой, перед всеми предыдущими).

Помимо служебной информации, заголовки письма также хранят и показываемую пользователю информацию, это обычно отправитель письма, получатель, тема и дата отправки.

Часто используемые поля

Return-Path — обратный адрес. Может отличаться от MAIL FROM (то есть обратный адрес может быть указан отличным от адреса отправителя).

Received — строчка журналирования прохождения письма. Каждый почтовый сервер (MTA) помечает процесс обработки этим сообщением. Если сообщение проходит через несколько почтовых серверов (обычная ситуация), то новые сообщения дописываются над предыдущими (и журнал перемещения читается в обратном порядке, от ближайшего узла к самому дальнему).

MIME-Version — версия MIME, с которым это сообщение создано. Поскольку сообщение создаётся раньше всех остальных событий с письмом, то этот заголовок обычно самый первый (то есть последний в списке).

From: — Имя и адрес отправителя (именно в этом заголовке появляется текстовое поле с именем отправителя). Может не совпадать с return-path и даже не совпадать с заголовком SMTP MAIL FROM:.

Sender: — Отправитель письма. Добавлено для возможности указать, что письмо от чьего-то имени (from) отправлено другой персоной (например, секретаршей от имени начальника). Некоторые почтовые клиенты показывают сообщение при наличии sender и from как «сообщение от 'sender' от имени 'from'». Sender является информационным заголовком (и также может отличаться от заголовка SMTP MAIL FROM).

To: — Имя и адрес получателя. Может содержаться несколько раз (если письмо адресовано нескольким получателям). Может не совпадать с полем SMTP RCPT TO.

cc: — (от англ. carbon copy). Содержит имена и адреса вторичных получателей письма, к которым направляется копия.

bcc: — (отангл. blind carbon copy).Содержит имена и адреса получателей письма, чьи адреса не следует показывать другим получателям. Это поле обычно обрабатывается почтовым сервером (и приводит к появлению нескольких разных сообщений, у которых bcc содержит только того получателя, кому фактически адресовано письмо). Каждый из получателей не будет видеть в этом поле других получателей из поля bcc.

Reply-To: — имя и адрес, куда следует адресовать ответы на это письмо. Если, например, письмо рассылается ботом, то в качестве Reply-To будет указан адрес персоны, готовой принять ответ на письмо.

Message-ID: — уникальный идентификатор сообщения. Состоит из адреса узла-отправителя и номера (уникального в пределах узла). Алгоритм генерации уникального номера зависит от сервера/клиента. Выглядит примерно так: AAB77AA2175ADD4BACECE2A49988705C0C93BB7B4A@example.com. Вместе с другими идентификаторами используется для поиска прохождения конкретного сообщения по журналам почтовой системы (почтовые системы фиксируют прохождение письма по его Message-ID) и для указания на письмо из других писем (используется для группировки и построения цепочек писем). Обычно создаётся первым почтовым сервером (MTA) в момент принятия почты от пользователя.

In-Reply-To: — указывает на Message-ID, для которого это письмо является ответом (с помощью этого почтовые клиенты могут легко выстраивать цепочку переписки — каждый новый ответ содержит Message-ID для предыдущего сообщения).

Subject: — тема письма.

Date: — дата написания письма.

Content-Type: — тип содержимого письма. С помощью этого поля указывается тип (HTML, RTF, Plain text) содержимого письма и кодировка, в которой создано письмо (см ниже про кодировки).

Помимо стандартных, почтовые клиенты, серверы и роботы обработки почты могут добавлять свои собственные заголовки, начинающиеся с «X-» (например, X-Mailer, X-MyServer-Note-OK или X-Spamassasin-Level).

Multipurpose Internet Mail Extensions (MIME) — стандарт, описывающий передачу различных типов данных по электронной почте, а также, шире, спецификация для кодирования информации и форматирования сообщений таким образом, чтобы их можно было пересылать по Интернету.

MIME определяет механизмы для передачи разного рода информации внутри текстовых данных (в частности, с помощью электронной почты), а именно: текст на языках, для которых используются кодировки, отличные от ASCII, и нетекстовый контент, такой как картинки, музыка, фильмы и программы. MIME является также фундаментальным компонентом коммуникационных протоколов, таких как HTTP, которым нужно, чтобы данные передавались в контексте сообщений подобных e-mail, даже если данные реально не являются e-mail.

Основной формат электронных сообщений определен в RFC 5322, который является обновленной версией RFC 2822 (который, в свою очередь, является обновленной версией RFC 822). Эти стандарты определяют похожие форматы для текстовых e-mail-заголовков и содержимого и правил, относящихся к общеиспользуемым полям, таким как «To:», «Subject:», «From:» и «Date:». MIME определяет набор e-mail-заголовков для определения дополнительных атрибутов сообщения, включая тип контента, и определяет множество кодировок, которые могут быть использованы для представления 8-битных бинарных данных, используя символы из 7-битного ASCII множества. MIME также определяет правила для кодирования не-ASCII символов в заголовках e-mail-сообщения, таких как «Subject:».

MIME расширяем для новых типов — его определение включает метод для регистрации новых типов контента и других атрибутов.

Тело письма

Тело письма отделяется от заголовка пустой строкой, а заканчивается (согласно стандартам SMTP) строчкой, состоящей из единственной точки (и символа перевода строки). Часть почтовых клиентов (например, Thunderbird) показывают эту точку, часть нет. В не-smtp стандартах формат письма зависит от стандарта системы (например, MAPI), но перед «выходом» письма за пределы MAPI-совместимой системы (например, перед пересылкой через Интернет) обычно приводится к SMTP-совместимому виду (иначе маршрутизация письма была бы невозможной, так как стандартом передачи почты в Интернете является SMTP).

Одним из существенных ограничений стандартов на почтовую пересылку является применение 7-битной кодировки (ASCII). Для английского текста это не представляет особой проблемы, однако, большинство неанглоязычных языков используют 8 (и более) битные кодировки, передача которых без искажений не гарантируется. Для целей совместимости, все не 7-битные кодировки приводятся в 7-битный вид (используя различные методы кодирования текста).

Пример тела письма после команды DATA:

Received: from alpha.bieberdorf.edu (alpha.bieberdorf.edu [124.211.3.11]) by mail.bieberdorf.edu (8.8.5) id 004A21; Tue, Mar 18 1997 14:36:17 -0800 (PST)

From: rth@bieberdorf.edu (R.T. Hood)

To: tmh@immense-isp.com

Date: Tue, Mar 18 1997 14:36:14 PST

Message-Id: <rth031897143614-00000298@mail.bieberdorf.edu>

X-Mailer: Loris v2.32

Subject: Lunch today?

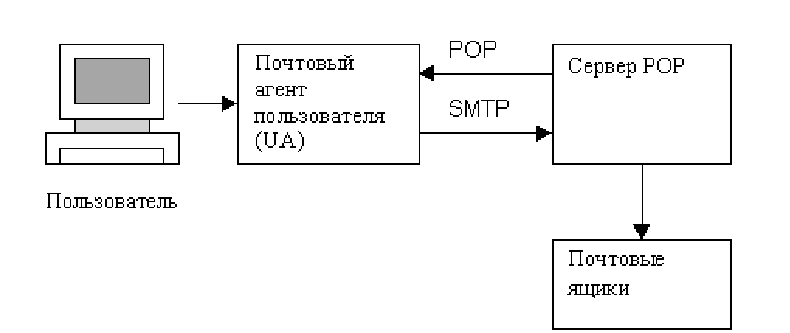
Do you have time to meet for lunch?

--rth

.

Протокол электронной почты POP3 РОРЗ (Post Office Protocol v.3) — это простейший протокол для работы пользователя с содержимым своего почтового ящика. Он позволяет только забрать почту из почтового ящика сервера на рабочую станцию клиента и удалить ее из почтового ящика на сервере. Всю дальнейшую обработку почтовое сообщение проходит на компьютере клиента.

Многие концепции, принципы и понятия протокола POP выглядят и функционируют подобно SMTP. Команды POP практически идентичны командам SMTP. На рис. 4.2 изображена модель взаимодействия клиента и сервера по протоколу POP. Сервер POP находится между агентом пользователя и почтовыми ящиками.

 Рис. 2.2 – Конфигурация модели клиент-сервер по протоколу POP3

В протоколе РОРЗ оговорены три стадии процесса получения почты:

авторизация, транзакция и обновление. После того как сервер и клиент РОРЗ 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). Наконец, на стадии обновления сеанс связи заканчивается.

Авторизация пользователя. После того как программа установила TCP-соединение с портом протокола РОРЗ (официальный номер 110), необходимо послать команду USER с именем пользователя в качестве параметра. Если ответ сервера будет +ОК, нужно послать команду PASS с паролем этого пользователя:

Пример:

CLIENT: USER ivan

SERVER: +ОК

CLIENT: PASS secret

SERVER: +ОК ivan's maildrop has 2 messages (320 octets)

Последняя строчка ответа означает, что в почтовом ящике ivan есть 2 сообщения (320 байтов).

Транзакции РОРЗ. После того как стадия авторизации окончена, обмен переходит на стадию транзакции. В следующих примерах демонстрируется возможный обмен сообщениями на этой стадии.

Команда STAT возвращает количество сообщений и количество байтов в сообщениях:

CLIENT: STAT

SERVER: +ОК 2 320

Команда LIST (без параметра) возвращает список сообщений в почтовом ящике и их размеры:

CLIENT: LIST

SERVER: +ОК

SERVER: 2 messages (320 octets)

SERVER: 1 120

SERVER: 2 200

...

Команда LIST с параметром возвращает информацию о заданном сообщении:

CLIENT: LIST 2

SERVER: +ОК 2 200

...

CLIENT: LIST 3

SERVER: -ERR no such message, only 2 messages in maildrop

Команда TOP возвращает заголовок, пустую строку и первые десять строк тела сообщения:

CLIENT: TOP 10

SERVER: +ОК

SERVER: <the POP3 server sends the headers of the message, a blank line, and the first 10 lines of the message body> (сервер POP высылает заголовки сообщений, пустую строку и первые десять строк тела сообщения)

SERVER:

....

CLIENT: TOP 100

SERVER: -ERR no such message

Команда NOOP не возвращает никакой полезной информации, за исключением позитивного ответа сервера. Однако позитивный ответ означает, что сервер находится в соединении с клиентом и ждет запросов:

CLIENT: NOOP

SERVER: +ОК

Следующие примеры показывают, как сервер POP3 выполняет действия.

Например, команда RETR извлекает сообщение с указанным номером и помещает его в буфер местного UA (почтового агента):

CLIENT: RETR 1

SERVER: +OK 120 octets

SERVER: <the POPS server sends the entire message here> (РОРЗ-сервер высылает сообщение целиком)

SERVER:

. . . . . .

Команда DELE отмечает сообщение, которое нужно удалить:

CLIENT: DELE 1

SERVER: +OK message 1 deleted ...

(сообщение 1 удалено) CLIENT:

DELE 2

SERVER: -ERR message 2 already deleted (сообщение 2 уже удалено)

Команда RSET снимает метки удаления со всех отмеченных ранее сообщений:

CLIENT: RSET

SERVER: +OK maildrop has 2 (в почтовом ящике 2 сообщения (320 байтов)) messages (320 octets)

Как и следовало ожидать, команда QUIT закрывает соединение с сервером:

CLIENT: QUIT

SERVER: +OK dewey POP3 server signing off

CLIENT: QUIT

SERVER: +OK dewey POP3 server signing off (maildrop empty)

CLIENT: QUIT

SERVER: +OK dewey POP3 server signing off (2 messages left)

Обратите внимание на то, что отмеченные для удаления сообщения на самом деле не удаляются до тех пор, пока не выдана команда QUIT и не началась стадия обновления. В любой момент в течение сеанса клиент имеет возможность выдать команду RSET, и все отмеченные для удаления сообщения будут восстановлены.

**Возможности реализации на различных языках программирования.**

Если вы выбрали в качестве языка программирования C#, то можно воспользоваться компонентом TcpClient. Который соответственно использует поток чтения и записи.

TcpClient tcpClient = new TcpClient();

//подключения по IP и по указанному номеру порта.

tcpClient.Connect("x.x.x.x", 9999);

//получение потока

networkStream = tcpClient.GetStream();

//поток для чтения

clientStreamReader = new StreamReader(networkStream);

//поток для записи

clientStreamWriter = new StreamWriter(networkStream);

while(true)

{

clientStreamReader.Read()

}

Очевидно, что необходимо проверять, когда поток окажется пустым или в соответствии с протоколом закончатся данные, что является следствием организации протокола TCP.

Для защищенного протокола SSL, типичный простой код без каких-либо проверок:

string server = "smtp.server.com";

TcpClient client = new TcpClient(server,25)

var stream = client.GetStream()

var sslStream = new SslStream(stream)

sslStream.AuthenticateAsClient(server);

var writer = new StreamWriter(sslStream)

var reader = new StreamReader(sslStream)

writer.WriteLine("EHLO " + server);

writer.Flush();

Console.WriteLine(reader.ReadLine());

Ниже приведен пример реализации соединения и передаче данных по протоколу ESMTP на python.

import socket

import ssl

from socket import \*

mailserver = 'smtp.mail.ru'

cSock = socket(AF\_INET, SOCK\_STREAM)

cSock.connect((mailserver, 465))

cSockSSL = ssl.wrap\_socket(cSock)

recv = cSockSSL.recv(1024)

print(recv)

cSockSSL.send("EHLO host\r\n")

recv = cSockSSL.recv(1024)

print(recv)

cSockSSL.close()

cSock.close()

**Для тех, кто будет использовать STARTTLS, можно воспользоваться помощью следующего кода. Естественно переработайте его.**

import ssl

from socket import \*

mailserver = 'smtp.mail.ru'

cSock = socket(AF\_INET, SOCK\_STREAM)

# лучше использовать 587, так как 25 используется во взаимодействии сервер-сервер. Хотя похоже работает и 25 (но это не точно).

# 587 используется для взаимодействия клиента с мейл трансфер агентом MTA.

cSock.connect((mailserver, 25))

cSock.send(b"EHLO host\r\n")

recv = cSock.recv(1024)

print(recv)

input()

cSock.send(b"STARTTLS\r\n")

recv = cSock.recv(1024)

print(recv)

input()

recv = cSock.recv(1024)

print(recv)

input()

# отключаем хендшейк при коннекте и используем tls, хотя можно и не отключать.

cSockSSL = ssl.wrap\_socket(cSock,ssl\_version=ssl.PROTOCOL\_TLS,do\_handshake\_on\_connect=False)

cSockSSL.do\_handshake()

cSockSSL.send(b"EHLO host\r\n")

recv = cSockSSL.recv(1024)

print(recv)

cSockSSL.send(b"AUTH LOGIN\r\n")

recv = cSockSSL.recv(1024)

print(recv)

cSockSSL.close()

cSock.close()

**Задание на лабораторную работу:**

Заведите почтовый ящик на каком-либо почтовом сервере, например, mail.ru. Осуществите подключение по протоколу TCP. Например, настройки почтового сервера smtp для mail.ru можно найти на <https://help.mail.ru/mail-help/mailer/popsmtp>. Соответственно, сервер smtp.mail.ru, порт 465, шифрование SSL/TLS. Описание формата mime https://www.opennet.ru/docs/RUS/mime\_rfc/.

Ознакомившись с протоколами SMTP, описанным в RFC 788 и POP3, описанным в RFC 1939 выполнить нижеприведенные задания по вариантам, необходимо создать как минимум два отдельных приложения.

Использовать сокеты, никаких готовых компонент.

1. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту. Реализовать так же консольное приложение для получения письма. Вытащить заголовки письма, если они есть.
2. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту. Добавить тему и дату письма. Реализовать так же консольное приложение для получения письма.
3. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту с mime формате в кодировке UTF-8. Добавить тему и дату письма. Реализовать так же консольное приложение для получения письма.
4. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту в виде html (использовать mime). Добавить тему и дату письма. Реализовать так же консольное приложение для получения письма.
5. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту в виде текста utf-8 (использовать mime), а также короткий звуковой файл. Реализовать так же консольное приложение для получения письма.
6. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту в виде текста utf-8 (использовать mime), а также короткий текстовый файл. Реализовать так же консольное приложение для получения письма.
7. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту в виде текста utf-8 (использовать mime), а также файл jpeg. Реализовать так же консольное приложение для получения письма.
8. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту. Получить текстовое сообщение и найти в нем boundary. Просмотреть список писем.
9. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту. Получить текстовое сообщение. Просмотреть список писем. Удалить письмо, просмотреть состояние почтового ящика.
10. Создать приложение, в котором будут вводиться текст письма, текст письма по протоколу SMTP должен быть отправлен на почту в виде текста utf-8 (использовать mime), а также файл png. Реализовать так же консольное приложение для получения письма.

Указанные приложения можно реализовать как в консольном варианте, так и с применением GUI. Для реализации можно использовать Java-Swing, AWT, C# - .Net , C++ - QT, Python.