Клиент протокола POP-3

Задание.

Разработать приложение для операционных систем семейства Windows, обеспечивающее функции клиента протокола POP-3.

Основные возможности.

Приложение должно реализовывать следующие функции:

1. Подключение к указанному серверу по IP-адресу или доменному имени
2. Получение состояния ящика (количество новых писем, их суммарная длина)
3. Получение списка заголовков всех новых писем сервера без предва- рительной загрузки
4. Загрузка всех новых или конкретных выбранных писем
5. Пометка конкретных писем для последующего удаления
6. Удаление помеченных писем
7. Выход из приложения без удаления помеченных писем
8. Подробное протоколирование соединения сервера с клиентом

Поддерживаемые команды.

Разработанное приложение должно реализовывать следующие команды протокола POP-3:

- USER – передача серверу идентификационной информации пользователя
- PASS – передача серверу пароля пользователя
- STAT – получение состояния почтового ящика
- LIST – получение списка сообщения почтового ящика
- RETR – получение сообщения
- DELE – пометка сообщения на удаление
- TOP – получение первых нескольких строк сообщения
- UIDL – получение уникального идентификатора сообщения
- RSET – сброс всех пометок на удаление сообщений
- QUIT – удаление всех помеченных сообщений и завершение сеанса

Настройки приложения.

Разработанное приложение должно предоставлять пользователю настройку следующих параметров:

1. IP-адрес или доменное имя почтового сервера
2. Номер порта сервера (по умолчанию - 110)
3. Имя пользователя
4. Пароль пользователя

Введение.

POP3 - стандартный интернет-протокол прикладного уровня, используемый клиентами электронной почты для получения почты с удаленного сервера по TCP/IP-соединению.

Базовые операции.

Первоначально, сервер прослушивает TCP соединение на порту 110. Когда клиент желает воспользоваться сервисом POP3, он должен установить соединение с сервером. После установки соединения сервер посылает клиенту приветствие. Клиент и POP3 сервер обмениваются командами и ответами до тех пор, пока соединение не будет закрыто или прервано.

Команды POP3 состоят из нечувствительного к регистру ключевого слова, за которым может следовать один или несколько аргументов. Все команды заканчиваются парой CRLF. Ключевые слова и аргументы состоят из печатаемых ASCII символов. Ключевые слова и аргументы разделены одиночным пробелом. Ключевые слова состоят из 3-х или 4-х символа, каждый аргумент может быть длиной до 40 символов.

Ответы состоят из индикатора состояния и ключевого слова, иногда сопровождаемого дополнительной информацией. Все ответы заканчиваются парой CRLF. Ответ может быть длиной до 512 символов, включая завершающий CRLF. В настоящее время есть два индикатора состояния: положительный (+OK) и отрицательный (-ERR).

Определенные ответы могут быть многострочными. В этом случае, после первой строки ответа заканчивающейся CRLF, каждая дополнительно посланная строка заканчивается парой CRLF. После того как все строки ответа посланы, последняя строка будет заканчиваться завершающим октетом — символом «.» и парой CRLF. Таким образом, многострочный ответ заканчивается этими пятью октетами — «CRLF.CRLF».

В протоколе POP3 предусмотрено 3 состояния сеанса:

1.**Авторизация:** После открытия клиентом TCP соединения, сервер посылает однострочное приветствие. Строка должна заканчиваться положительным ответом.

Пример:

```
S:  +OK POP3 server ready
```

Теперь сессия находится в состоянии AUTHORIZATION. Клиент должен идентифицировать себя на сервере.

Для идентификации с помощью команд USER и PASS, клиент должен сначала послать команду USER. Если сервер ответил положительным индикатором состояния (+OK), то клиент должен послать команду PASS чтобы закончить авторизацию или послать команду QUIT для

завершения сессии. Если сервер отправил отрицательный ответ (-ERR) на команду USER, то можно повторить авторизацию или закончить сессию командой QUIT.

2.**Транзакция:**

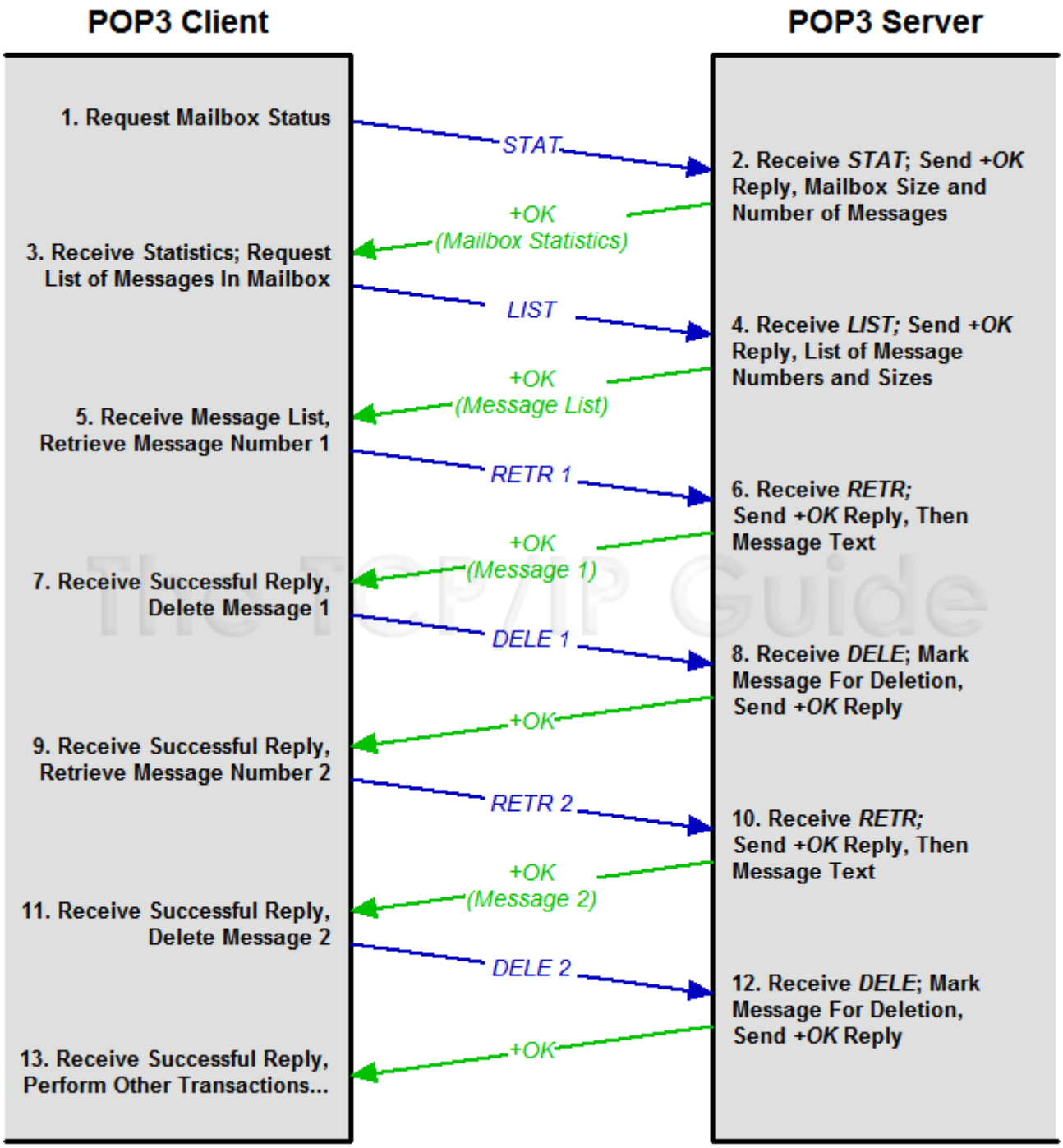
После того как клиент успешно идентифицировал себя на сервере и сервер заблокировал и открыл соответствующий почтовый ящик, сессия переходит в состояние TRANSACTION. Теперь клиент может запрашивать информацию. После каждой команды сервер отправляет ответ. В конце клиент отправляет команду QUIT, и сессия переходит в состояние UPDATE.

3.**Обновление:**

Когда клиент посылает команду QUIT в состоянии TRANSACTION, сервер переходит на стадию UPDATE.

Если сессия завершается по каким-либо другим причинам, без посылки команды QUIT, POP3 сессия не входит в стадию UPDATE и ни одно сообщение из почтового ящика не должно быть удаленно.

Пример взаимодействия сервера и клиента в виде диаграмм последовательностей.



Пример POP3 сессии

```
S: <wait for connection on TCP port 110>
C: <open connection>
S:    +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C:    APOP mrose c4c9334bac560ecc979e58001b3e22fb
S:    +OK mrose's maildrop has 2 messages (320 octets)
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
```

```
S:    +OK 120 octets
S:    <the POP3 server sends message 1>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <the POP3 server sends message 2>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

Информация взята из RFC 1939 https://www.ietf.org/rfc/rfc1939.txt (перевод на русский язык http://www.vanderboot.ru/rfc/mail/1939.php)

Архитектура приложения

Поддерживаемые команды.

- USER – передача серверу идентификационной информации пользователя
- PASS – передача серверу пароля пользователя
- STAT – получение состояния почтового ящика
- LIST – получение списка сообщения почтового ящика
- RETR – получение сообщения
- DELE – пометка сообщения на удаление
- TOP – получение первых нескольких строк сообщения
- UIDL – получение уникального идентификатора сообщения
- RSET – сброс всех пометок на удаление сообщений
- QUIT – удаление всех помеченных сообщений и завершение сеанса

Классы и методы

В приложении реализован главный класс протокола - POP3 и класс POP3_SSL, являющийся наследником POP3, а также реализующий шифрование по средством SSL. Класс Client является непосредственно клиентом протокола POP3, предоставляющим консольный интерфейс.

Названия функции класса POP3, которые реализуют основную функциональность протокола совпадают с поддерживаемыми командами, их можно использовать при создании альтернативной оболочки клиента, например с графическим интерфейсом.

Класс POP3

Методы:

- _create_socket(self, timeout) - создание сокета для открытия соединения
- _putcmd(self, line) - отправка команды
- _getline(self) - чтение принятых данных из сокета
- _getresp(self) - получение ответа от сервера
- _getlongresp(self) - получение "длинного" ответа от сервера (например включающего в себя список писем)
- _shortcmd(self, line) - отправка команды без аргумента
- _longcmd(self, line) - отправка команды с аргументом (например номером сообщения для последующей загрузки)
- close(self) - закрытие соединения

Далее следуют функции, обрабатывающие согласно протоколу одноименные команды от клиента:

- user(self, user):
- pass_(self, pswd):
- stat(self):
- list(self, which):
- retr(self, which):
- dele(self, which):
- rset(self):
- quit(self):
- top(self, which, howmuch):
- uidl(self, which=None):

Класс Client

Все методы данного класса служат для предоставления пользователю консольного интерфейса для использования основных функций протокола POP3.

Методы:

- del_msg(self) - удаление сообщения
- print_top(self) - вывод нескольких первых сообщений
- print_list(self) - вывод списка сообщений
- print_stat(self) - вывод статистики почтового ящика
- print_message1(self) - вывод сообщения
- print_uidl(self) - вывод уникального идентификатора сообщения
- send_user(self) - отправка имени пользователя на сервер
- send_pass(self) - отправка пароля на сервер

Тестирование

Тестирование данного приложения проводилось на примере его взаимодействия с почтовым сервером Яндекс.

```
Enter pop server:1
Enter pop port:1
resp from server: b'+OK POP Ya! na@6 BORhUH1KViEt'
Enter cmd:uni
resp from server: b'+OK password, please.'
resp from server: b'+OK 4 6490'
Enter cmd:retr
enter num of msg: 1
resp from server: b'+OK 1618 octets.'
Message:Test4
Enter cmd:uidl
enter num of msg or press enter:
resp from server: b'+OK 4 6490'
b'1 a703c7e378807083bc4158ca152cfc85'
b'2 6670eba6e63f9d8c4ba9a065fb119c4b'
b'3 b59c8a9cac8d1d48b362bc6e073f46b3'
b'4 18833eae8c9d63400f1a9d859f76adc6'
Enter cmd:list
enter num of msg or press enter:
resp from server: b'+OK 4 6490'
    b'1 1618'
b'2 1627'
b'3 1618'
b'4 1627'
Enter cmd:quit
resp from server: b'+OK shutting down.'
```

Также были написаны тесты, проверяющие основные команды. Был использован "мок", который подменяет ответ сервера на запрос о сообщении. При запуске тестов, было выявлено, что они покрывают 100% файлов и 67% строк кода.

Исходные коды тестов приведены в Приложении №2.

Приложение №1. Код программы.

```
from email.parser import FeedParser
import errno
import re
import socket

try:
    import ssl
    HAVE_SSL = True
except ImportError:
    HAVE_SSL = False

POP3_PORT = 110
POP3_SSL_PORT = 995
CR = b'\r'
LF = b'\n'
CRLF = CR+LF
```

```
_MAXLINE = 2048



class POP3:

    encoding = 'UTF-8'

    def __init__(self, host, port=POP3_PORT,
                 timeout=socket._GLOBAL_DEFAULT_TIMEOUT):
        self.host = host
        self.port = port
        self.sock = self._create_socket(timeout)
        self.file = self.sock.makefile('rb')
        self.welcome = self._getresp()

    def _create_socket(self, timeout):
        return socket.create_connection((self.host, self.port), timeout)

    def _putcmd(self, line):
        line = bytes(line, self.encoding)
        self.sock.sendall(line + CRLF)

    def _getline(self):
        line = self.file.readline(_MAXLINE + 1)
        octets = len(line)
        return line[:-2], octets

    def _getresp(self):
        resp, o = self._getline()
        print('resp from server: ' + str(resp))
        return resp

    def _getlongresp(self):
        resp = self._getresp()
        list = []; octets = 0
        line, o = self._getline()
        while line != b'.':
            if line.startswith(b'..'):
                o = o-1
                line = line[1:]
            octets = octets + o
            list.append(line)
            line, o = self._getline()
        return resp, list, octets

    def _shortcmd(self, line):
        self._putcmd(line)
        return self._getresp()

    def _longcmd(self, line):
        self._putcmd(line)
        return self._getlongresp()

    def user(self, user):
        return self._shortcmd('USER %s' % user)
```

```
    def pass_(self, pswd):
        return self._shortcmd('PASS %s' % pswd)


    def stat(self):
        retval = self._shortcmd('STAT')
        rets = retval.split()
        numMessages = int(rets[1])
        sizeMessages = int(rets[2])
        return (numMessages, sizeMessages)


    def list(self, which=None):
        if which is not None:
            return str(self._shortcmd('LIST %s' % which))
        return self._longcmd('LIST')

    def retr(self, which):
        return self._longcmd('RETR %s' % which)

    def dele(self, which):
        return self._shortcmd('DELE %s' % which)

    def rset(self):
        return self._shortcmd('RSET')


    def quit(self):
        resp = self._shortcmd('QUIT')
        self.close()
        return resp

    def close(self):
        if self.file is not None:
            self.file.close()
        if self.sock is not None:
            try:
                self.sock.shutdown(socket.SHUT_RDWR)
            except OSError as e:
                if e.errno != errno.ENOTCONN:
                    raise
            finally:
                self.sock.close()
        self.file = self.sock = None

    timestamp = re.compile(br'\+OK.*(<[^>]+>)')

    def top(self, which, howmuch):
        return self._longcmd('TOP %s %s' % (which, howmuch))

    def uidl(self, which=None):
        if which is not None:
            return self._shortcmd('UIDL %s' % which)
        return self._longcmd('UIDL')

    def uni_cmd(self):
        f = open('login.txt','r')
```

```
        for line in f:
            usr = line.split(' ')[0]
            paswd = line.split(' ')[1]
        self.user(usr)
        self.pass_(paswd)

if HAVE_SSL:

    class POP3_SSL(POP3):

        def __init__(self, host, port=POP3_SSL_PORT, keyfile=None, certfile=None,
                     timeout=socket._GLOBAL_DEFAULT_TIMEOUT, context=None):
            if context is not None and keyfile is not None:
                raise ValueError("context and keyfile arguments are mutually "
                                 "exclusive")
            if context is not None and certfile is not None:
                raise ValueError("context and certfile arguments are mutually "
                                 "exclusive")
            self.keyfile = keyfile
            self.certfile = certfile
            if context is None:
                context = ssl._create_stdlib_context(certfile=certfile,
                                                     keyfile=keyfile)
            self.context = context
            POP3.__init__(self, host, port, timeout)

        def _create_socket(self, timeout):
            sock = POP3._create_socket(self, timeout)
            sock = self.context.wrap_socket(sock,
                                            server_hostname=self.host)
            return sock

    class Client:

        def __init__(self, serv, port):
            self.serv = serv
            self.port = port
            self.pop3 = POP3_SSL(self.serv, self.port)

        def del_msg(self):
            num = input('Enter number of msg: ')
            self.pop3.dele(int(num))

        def print_top(self):
            msg = input('number of msg:')
            num = input('number of rows:')
            r,l,o=self.pop3.top(int(msg),int(num))
            print('%s' % (str(l)))

        def print_list(self):
            num = input('enter num of msg or press enter: ')
            if num == '':
                r,l,o=self.pop3.list()
                for i in l:
                    print(str(i))
            else:
                r = self.pop3.list(int(num))
```

```
                print(r)

        def print_stat(self):
            num,size=self.pop3.stat()
            print('num=%s, szie=%s' % (str(num), str(size)))

        def print_message1(self):
            num = input('enter num of msg: ')
            tmp = self.pop3.retr(num)
            for i in tmp:
                print(str(i))
            raw = str(tmp[1])
            mes = raw.split(',')
            l = len(mes)
            message = mes[l-7]
            if str(message)[3:-1] != 'Content-Transfer-Encoding: 7bit':
                print ('Message:' + str(message)[3:-1])
            else:
                message = mes[l-2]
                print ('Message:' + str(message)[3:-1])

        def print_uidl(self):
            num = input('enter num of msg or press enter: ')
            if num == '':
                r,l,o=self.pop3.uidl()
                for i in l:
                    print('%s' % str(i))
            else:
                r=self.pop3.uidl(int(num))
                print(r)

        def send_user(self):
            userresp = ''
            while not userresp.startswith('b\'+OK'):
                user = input('Enter your login:')
                userresp = str(self.pop3.user(user))

        def send_pass(self):
            passresp = ''
            while not passresp.startswith('b\'+OK'):
                passwd = input('Enter your password:')
                passresp = str(self.pop3.pass_(passwd))

if __name__ == "__main__":

    serv = input('Enter pop server:')
    if serv == '1':
        serv = 'pop.yandex.ru'
    port = input('Enter pop port:')
    if port == '1':
        port = '995'

    cli = Client(serv, port)

    cds = {
        'uni' : cli.pop3.uni_cmd,
        'user' : cli.send_user,
```

```
        'pass' : cli.send_pass,
        'stat' : cli.print_stat,
        'retr' : cli.print_message1,
        'list' : cli.print_list,
        'top' : cli.print_top,
        'uidl' : cli.print_uidl,
        'quit' : cli.pop3.quit,
        'dele' : cli.del_msg,
        'rset' : cli.pop3.rset,
        }
    cmd = ''

    while cmd != 'quit':
        cmd = input('Enter cmd:')
        if cmd in cds:
            cds[cmd]()
        else:
            print('Wrong cmd!')
```

Приложение №2. Код тестов.

```
import unittest
from mock import patch, Mock
from client import POP3_SSL


class testPOP3(unittest.TestCase):

    def setUp(self):

        f = open('login.txt','r')
        for line in f:
            self.login = line.split(' ')[0]
            self.passwd = line.split(' ')[1]
        f.close()
        self.serv = 'pop.yandex.ru'
        self.port = '995'
        self.client = POP3_SSL(self.serv,self.port)
        self.client.user(self.login)
        self.client.pass_(self.passwd)

    def tearDown(self):
        self.client.quit()

    def test_no_retr(self):
        self.assertEqual(self.client.retr(2)[0].startswith(b'-ERR'), False)

    def get_mes(self, resp):
        tmp = resp
        raw = str(tmp[1])
        mes = raw.split(',')
        l = len(mes)
        message = mes[0]
        return message
```

```
    def test_retr(self):
        self.assertNotEqual(self.get_mes(self.client.retr(1))," b\'Smtng wrong'")

    def test_stat(self):
        self.assertGreaterEqual(self.client.stat(), (0,0))

    def test_dele(self):
        tmp1 = self.client.stat()[0]
        self.client.dele(1)
        self.client.quit()
        print('Before dele: %s' % str(tmp1))
        self.client = POP3_SSL(self.serv,self.port)
        self.client.user(self.login)
        self.client.pass_(self.passwd)
        tmp2 = self.client.stat()[0]
        print('After dele: %s' % str(tmp2))
        self.assertLessEqual(tmp2, tmp1)


    def test_rset(self):
        tmp1 = self.client.stat()[0]
        self.client.dele(1)
        self.client.rset()
        self.client.quit()
        self.client = POP3_SSL(self.serv,self.port)
        self.client.user(self.login)
        self.client.pass_(self.passwd)
        tmp2 = self.client.stat()[0]
        self.assertEqual(tmp2, tmp1)

    def test_call_retr(self):
        with patch('client.POP3.retr') as mock:
            instance = mock.return_value
            instance.method.return_value = b'fdfdf'
            result = testPOP3.test_retr(self)
            mock.assert_called_once_with(1)

if __name__ == '__main__':
    unittest.main()
```