## Лабораторная работа №3 Работа с HTTP и FTP

Лабораторная работа №3 выполняется после изучения материала, посвященного описанию принципов использования C#, Java, Python, C++ для управления соединениями с сервером по протоколам HTTP и FTP.

Цель работы:

Написать приложения для ОС Windows или OC Linux, представляющие собой простые HTTP и FTP – клиенты. Для отправки команд нужно использовать сокетные соединения, для парсинга можно использовать готовые компоненты. Лабораторная считается сделанной, если создано подключение и возможно скачать файл FTP или сделать просмотр файлов и каталогов, для HTTP или HTTPS можно запросить html страницу, в ней найти ссылку и скачать какой-либо объект по этой ссылке.

Приложение FTP (FTPS, rfc4217, https://tools.ietf.org/html/rfc4217) клиента использует указанные протоколы. Не путайте FTPS (SSL FTP) c FTP через SSH (туннелирование обычного FTP через соединение SSH, защита реализуется только для пассивного клиентского соединения из-за особенности работы FTP в активном режиме требующего подключения сервера к клиенту). Так же есть SFTP (SHH FTP, защищенное через SHH FTP соединение, SSH2, отдельный от FTP протокол, продолжение SSH), а также с SFTP (Simple FTP, урезанная версия FTP).

Желающие могут сделать GUI десктоп приложение. Например, используя C# и Visual Studio это сделать достаточно просто, путем размещения соответствующих компонент ввода текстовых полей, кнопок на форме и задав им соответствующие события, и используя свойства данных компонент.

Использование QT требует бОльших затрат на изучение данной технологии, но может дать неоценимый опыт при изучении концепции использования сигналов и слотов, при реализации возможностей обработки событий. Для создания приложений на QT рекомендуется использовать QTCreator и QT Designer. GUI приложения на Java могут использовать swing, апплеты awt, библиотека javafx.

Очень простой пример на Python c использованием wxPython. Так же может понадобиться установка python-wxgtk-webview3.0 под Linux. Здесь реализована одна простая функция перехода по ссылке, при этом нет обработки события открытия нового окна, назад, вперед и многих других функций.

import wx

import wx.html2

class TestFrame(wx.Frame):

go = []

address = []

browser = []

#обработчик события нажатия кнопки

def onGoButton(self,event):

print("ButtonClick")

print(self.address.GetValue())

#загрузка страницы, адрес берется из строки ввода

self.browser.LoadURL(self.address.GetValue())

return

def \_\_init\_\_(self, parent, title):

wx.Frame.\_\_init\_\_(self, parent, id=-1, title=title)

text = wx.StaticText(self, label=title)

# компонент бокссайзер позволяет управлять размещением объектов на форме

#(в данном случае вертикально)

sizer = wx.BoxSizer(wx.VERTICAL)

#объект для создания минибраузера

self.browser = wx.html2.WebView.New(self)

#объект для текстовой строки ввода

self.address = wx.TextCtrl(self, value="http://")

#объект кнопка

self.go = wx.Button(self, label="Go", id=wx.ID\_OK)

#добавляем объекты для пропорционального размещения в соотношении 90, 5, 5

sizer.Add(self.browser, proportion = 90, flag = wx.EXPAND, border = 10)

sizer.Add(self.address, proportion = 5, flag = wx.EXPAND, border = 10)

sizer.Add(self.go, proportion = 5, flag = wx.EXPAND, border = 10)

self.SetSizer(sizer)

self.SetSize((1000, 800))

#связываем обработчик события onGoButton и кнопку go

self.Bind(wx.EVT\_BUTTON,self.onGoButton,self.go)

app = wx.App()

frame = TestFrame(None, "Simple browser")

frame.Show()

app.MainLoop()

Пример ftp клиента на python, показывающего некоторые компоненты, которые можно использовать при выполнении задания, нужно проверить на наличие блокировок, добавить по заданию нужные действия, можно делать на другом языке программирования. Пример работает не на всех серверах, Вы должны это исправить, в том числе для вашего варианта, чтобы на всех серверах работало, так, например, не учтен первый прием данных с кодом xyz-, не учтен пробел в конце кода xyz, не учтено что может быть получен пустой буфер, не учтено что код xyz может находиться на последней строке полученного буфера.

Сервера которые вы можете использовать и на которых должно работать ваше приложение:

1. test.rebex.net (логин demo, пароль password)
2. [ftp.gnu.org](ftp://ftp.gnu.org) (логин anonymous, пароль anonymous)
3. [ftp.pureftpd.org](ftp://ftp.pureftpd.org)
4. [ftp.vim.org](ftp://ftp.vim.org)
5. [ftp.slackware.com](ftp://ftp.slackware.com)
6. in1.hostedftp.com

import socket

import re

import time

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

res = s.connect(("url to server", 21))

print("Connection OK")

except:

print("Error connection")

buf = s.recv(1024)

print(buf.decode('utf-8'))

CMD\_USER = b"USER anonymous\r\n"

CMD\_PASS = b"PASS anonymous@anon.org\r\n"

CMD\_PASV = b"PASV\r\n"

CMD\_LIST = b"LIST\r\n"

CMD\_QUIT = b"QUIT\r\n"

def send\_cmd(cmd: bytes):

time.sleep(0.1)

s.send(cmd)

print(cmd[:-2].decode('utf-8'))

buf = s.recv(1024)

print(buf.decode('utf-8'))

# Если многострочный прием, по rfc начинается с xyz- заканчивается xyz<SP>

if buf[3] == ord("-"):

res = buf[0:3].decode('utf-8')

while(True):

buf = s.recv(1024)

print(buf.decode('utf-8'))

if(buf[0:3].decode()==res):

break

# Если требуется продолжение действий (читать rfc по кодам ответов ftp)

if(buf[0]== ord("1")):

buf = s.recv(1024)

print(buf.decode('utf-8'))

return buf[:-2].decode('utf-8')

send\_cmd(CMD\_USER)

send\_cmd(CMD\_PASS)

ret = send\_cmd(CMD\_PASV)

# ищем подстроку соответствующую регулярному выражению

match = re.search(r"(\d+,\d+,\d+,\d+,\d+,\d+)",ret)

# разбиваем через запятую получая список чисел соответствующих ip port

match = re.split(r",",match[0])

ip = ".".join(match[0:4])

port = int(match[4])\*256+int(match[5])

print(f"ip {ip} port {port}")

s1 = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

res = s1.connect((ip, port))

print("Connection OK")

except:

print("Error connection")

send\_cmd(CMD\_LIST)

buf = s1.recv(1024)

print(buf)

send\_cmd(CMD\_QUIT)

s.close()

Пример HTTP. Данный пример показывает подключение к веб серверу и получение html страницы. Затем парсит ее. Пример приведен с целью показать основные компоненты, которые можно использовать. Если необходим https, можно воспользоваться примером из второй лабораторной с использованием ssl.

from socket import \*

http\_server = 'example.org'

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

cSock.connect((http\_server, 80))

cSock.send(b"GET /index.html HTTP/1.0\r\n")

cSock.send(b"HOST: example.org\r\n")

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

from http\_parser.http import HttpStream

from http\_parser.reader import SocketReader

r = SocketReader(cSock)

p = HttpStream(r)

print(p.headers())

body = p.body\_file().read()

from html.parser import HTMLParser

class tHTMLParser(HTMLParser):

def handle\_starttag(self, tag, attrs):

print("Start tag:", tag)

for attr in attrs:

print(" attr:", attr)

parser = tHTMLParser()

parser.feed(body.decode('utf8'))

Общее задание. Создать простейший ftp и http клиент. Обеспечить подключение к серверу и выполнение команд по заданию.

Варианты заданий.

1. FTP клиент должен просматривать файлы, переходить в другую папку, скачивать файл. HTTP должен выдать заголовки запроса, перейти по ссылке и скачать документ находящийся по этой ссылке.
2. FTP клиент должен просматривать файлы, выдавать размера файла, переходить в другую папку, скачивать файл. HTTP должен выдать заголовки запроса, перейти к изображению и скачать файл находящийся по ссылке.
3. FTP клиент должен просматривать файлы, переименовать файл, переходить в другую папку, скачивать файл. HTTP должен выдать заголовки запроса, перейти к ссылке и скачать файл html находящийся по ссылке, выдать титульные заголовки html.
4. FTP клиент должен просматривать файлы, закачать файл, переходить в другую папку, скачивать файл. HTTP должен выдать заголовки запроса, перейти к ссылке и скачать файл html находящийся по ссылке, выдать титульные заголовки html.
5. FTP клиент должен просматривать файлы, закачать файл, переходить в другую папку, скачивать файл. HTTP должен выдать заголовки запроса, перейти к ссылке и скачать файл html находящийся по ссылке, выдать титульные заголовки html.
6. FTP клиент должен просматривать файлы, просматривать время модификации, переходить в другую папку, скачивать файл. HTTP должен выдать заголовки запроса, перейти к ссылке и скачать файл html находящийся по ссылке, выдать список заголовков ответа, и найти все img ссылки.
7. FTP клиент должен просматривать файлы в кратком формате, просматривать время модификации, переходить в другую папку, скачивать файл. HTTP должен выдать заголовки запроса, перейти к ссылке и скачать файл html находящийся по ссылке, выдать весь список тэгов.
8. FTP клиент должен просматривать файлы в кратком формате, создать каталог, удалить каталог, переходить в другую папку. HTTP должен выдать заголовки запроса, перейти к ссылке и скачать файл html находящийся по ссылке, выдать весь список тэгов.
9. FTP клиент должен просматривать файлы в кратком формате и подробном формате, удалить каталог, переходить в другую папку. HTTP должен выдать заголовки запроса, перейти к ссылке и скачать файл html находящийся по ссылке, выдать А теги.
10. FTP клиент должен просматривать файлы и каталоги и скачать два файла, переходить в другую папку. HTTP должен выдать заголовки запроса, перейти по ссылке и скачать файл html находящийся по ссылке, выдать его img теги.