Programming Assignment: Сервер для приема метрик

You have not submitted. You must earn 1/1 points to pass.

Deadline Pass this assignment by April 8, 12:59 AM PDT

Instructions My submission Discussions

На предыдущей неделе вы разработали клиентское сетевое приложение — клиента для сервера метрик, который умеет отправлять и получать всевозможные метрики. Пришло время финального задания — нужно реализовать серверную часть самостоятельно.

Как обычно вам необходимо разработать программу в одном файле-модуле, который вы загрузите на проверку обычным способом. Сервер должен соответствовать протоколу, который был описан в задании к предыдущей неделе. Он должен уметь принимать от клиентов команды **put** и **get**, разбирать их, и формировать ответ согласно протоколу. По запросу put требуется сохранять метрики в структурах данных в памяти процесса. По запросу get сервер обязан отдавать данные в правильной последовательности.

На верхнем уровне вашего модуля должна быть объявлена функция **run_server(host, port) —** она принимает адрес и порт, на которых должен быть запущен сервер.

Для проверки правильности решения мы воспользуемся своей реализацией клиента и будем отправлять на ваш сервер put и get запросы, ожидая в ответ правильные данные от сервера (согласно объявленному протоколу). Все запросы будут выполняться с таймаутом — сервер должен отвечать за приемлемое время.

Сервер должен быть готов к неправильным командам со стороны клиента и отдавать клиенту ошибку в формате, оговоренном в протоколе. В таком случае работа сервера не должна завершаться аварийно.

На последней неделе мы с вами разбирали пример tcp-сервера на asyncio:

```
import asyncio


class ClientServerProtocol(asyncio.Protocol):
    def connection_made(self, transport):
        self.transport = transport

    def data_received(self, data):
        resp = process_data(data.decode())
        self.transport.write(resp.encode())


loop = asyncio.get_event_loop()
coro = loop.create_server(
    ClientServerProtocol,
    '127.0.0.1', 8181
)

server = loop.run_until_complete(coro)

try:
    loop.run_forever()
except KeyboardInterrupt:
    pass

server.close()
loop.run_until_complete(server.wait_closed())
loop.close()
```

Данный код создает tcp-соединение для адреса 127.0.0.1:8181 и слушает все входящие запросы. При подключении клиента будет создан новый экземпляр класса ClientServerProtocol, а при поступлении новых данных вызовется метод этого объекта - data_received. Внутри asyncio.Protocol спрятана вся магия обработки запросов через корутины, остается реализовать протокол взаимодействия между клиентом и сервером.

Этот код может использоваться как основа для реализации сервера. Это не обязательное требование. Для реализации задачи вы можете использовать любые вызовы из стандартной библиотеки Python 3. Сервер должен обрабатывать запросы от нескольких клиентов одновременно.

В процессе разработки сервера для тестирования работоспособности вы можете использовать клиент, написанный на предыдущей неделе.

Давайте еще раз посмотрим на текстовый протокол в действии при использовании утилиты telnet:

```
$: telnet 127.0.0.1 8888
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
> get test_key
< ok
<
> got test_key
< error
< wrong command
<
> put test_key 12.0 1503319740
< ok
<
> put test_key 13.0 1503319739
< ok
<
> get test_key
< ok
< test_key 13.0 1503319739
< test_key 12.0 1503319740
<
> put another_key 10 1503319739
< ok
<
> get *
< ok
< test_key 13.0 1503319739
< test_key 12.0 1503319740
< another_key 10.0 1503319739
<
```

Также вы можете воспользоваться вспомогательным скриптом, который использует "'эталонную" реализацию клиента, открывающуюся после сдачи задания на пятой неделе, для локального тестирования написанного вами сервера:

```
"""
Это вспомогательный скрипт для тестирования сервера из задания на
  неделе 6.

Для запуска скрипта на локальном компьютере разместите рядом файл
  client.py,
где содержится код клиента, который открывается по прохождении задания
недели 5.

Сначала запускаете ваш сервер на адресе 127.0.0.1 и порту 8888, а затем

запускаете этот скрипт.
"""
import sys
from client import Client, ClientSocketError, ClientProtocolError


def run(host, port):

    client1 = Client(host, port, timeout=5)
    client2 = Client(host, port, timeout=5)

    try:
        client1.connection.sendall(b"malformed command test\n")
        client1._read()
        client2.connection.sendall(b"malformed command test\n")
        client2._read()
    except ClientSocketError as err:
        print(f"Ошибка общения с сервером: {err.__class__}: {err}")
        sys.exit(1)
    except ClientProtocolError:
        pass
    else:
        print("Неверная команда, отправленная серверу, должна
          возвращать ошибку протокола")
        sys.exit(1)

    try:
        client1.put("k1", 0.25, timestamp=1)
        client2.put("k1", 2.156, timestamp=2)
        client1.put("k1", 0.35, timestamp=3)
        client2.put("k2", 30, timestamp=4)
        client1.put("k2", 40, timestamp=5)
        client1.put("k2", 40, timestamp=5)
    except Exception as err:
        print(f"Ошибка вызова client.put(...) {err.__class__}: {err}")
        sys.exit(1)

    expected_metrics = {
        "k1": [(1, 0.25), (2, 2.156), (3, 0.35)],
        "k2": [(4, 30.0), (5, 40.0)],
    }

    try:
        metrics = client1.get("*")
        if metrics != expected_metrics:
            print(f"client.get('*') вернул неверный результат.
              Ожидается: {expected_metrics}. Получено: {metrics}")
            sys.exit(1)
    except Exception as err:
        print(f"Ошибка вызова client.get('*') {err.__class__}: {err}")
        sys.exit(1)

    expected_metrics = {"k2": [(4, 30.0), (5, 40.0)]}

    try:
        metrics = client2.get("k2")
        if metrics != expected_metrics:
            print(f"client.get('k2') вернул неверный результат.
              Ожидается: {expected_metrics}. Получено: {metrics}")
            sys.exit(1)
    except Exception as err:
        print(f"Ошибка вызова client.get('k2') {err.__class__}: {err}")
        sys.exit(1)

    try:
        result = client1.get("k3")
        if result != {}:
            print(f"Ошибка вызова метода get с ключом, который еще не
              был добавлен. Ожидается: пустой словарь. Получено:
              {result}")
            sys.exit(1)
    except Exception as err:
        print(f"Ошибка вызова метода get с ключом, который еще не был
          добавлен: {err.__class__} {err}")
        sys.exit(1)

    print("Похоже, что все верно! Попробуйте отправить решение на
      проверку.")


if __name__ == "__main__":
    run("127.0.0.1", 8888)
```

Успехов в разработке!

How to submit

When you're ready to submit, you can upload files for each part of the assignment on the "My submission" tab.