Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação

Redes de Computadores I (CK0249) 2021.1 - PPE
Prof. Dr. Emanuel Bezerra Rodrigues

ATIVIDADE PRÁTICA I
PROGRAMAÇÃO COM SOCKETS

Aluna: Fernanda Costa de Sousa
Matrícula: 485404

RESUMO

Este relatório contém uma explicação sussinta sobre o servidor HTTP referente a atividade de programação com sockets, acerca dos códigos e seu funcionamento, prints dos respectivos códigos, link para youtube onde faço uma demonstraçao do funcionamento do servidor além do link para o Github onde o código se encontra.

SERVIDOR HTTP

Nessa tarefa optei por desenvolver um servidor simples em Python que atendesse os requisitos especificados. O servidor cria o socket de conexão, recebe a requisição HTTP dessa conexão. Analisa a requisição, obtém o arquivo requisitado cria uma mensagem de resporta HTTP com linhas de cabeçalho e o arquivo requisitado e envia a resporta por uma conexão TCP ao navegador que requisitou. Também é retornada uma mensagem de erro “404 Not found”.

EXPLICANDO O CÓDIGO

Inicialmente fiz a importação da biblioteca socket.

```
import socket
```

Em seguida fiz a definição do host e da porta do servidor. No host o ip está em branco e a porta escolhida por mim foi a 8081.

```
HOST = ''
PORT = 8081
```

Na linha 9 criei um socket com Ipv4 (AF_INET usando TCP (SOCK_STREAM).

```
listen_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Na linha 11 eu tenho esse comando que permite que seja possível reusar o endereço e porta do servidor caso ele seja encerrado incorretamente.

```
listen_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

Na linha13 o listen_socket.bind faz o vínculo entre o socket e a porta, faz o bind do IP do servidor com a porta.

```
listen_socket.bind((HOST, PORT))
```

O listen_socket.listen() escuta pedidos na porta do socket do servidor.

```
listen_socket.listen(1)
```

Na linha 17 eu to impriminto que o servidor está pronto para receber novas conexões.

```
print ('Serving HTTP on port %s ...' % PORT)
```

No laço (While True) ele fica aguardando por novas conexões. O método .recv recebe os daos enviados por um cliente através do socket, depois imprime na tela o que o cliente enviou ao servidor.
Nessa parte eu especifico como ele deve receber a requisição HTTP dessa conexão. Ele faz a análise da requisição para determinar o arquivo sendo requisitado.

```
client_connection, client_address = listen_socket.accept()
request = client_connection.recv(1024).decode()
request = request
request = request.split(" ")
```

Em seguida ele cria uma mensagem de resposta HTTP consistindo no arquivo requisitado e também as linhas de cabeçalho com o protocolo, a versão e a mensagem de sucesso que é o 200. Caso o que foi requisitado seja o metodo GET precedido de barra (/), então por default eu associo o index.html como arquivo. Então ele envia a resposta pela conexão TCP ao navegador que fez a requisição.

```
if(request[0] == 'GET'):
file = request[1].split(" ")
print(file[0][1:])
if(file[0] == "/"):
index = open('index.html')
contend = index.read()
http_response = """HTTP/1.1 200 OK\r\n\r\n""" + str(contend)
index.close()
```

Eu fiz também um tratamento para o caso em que uma requisição de um arquivo que não existe por exemplo, ele ser capaz se retorna uma mensagem de erro 404 not found e uma página em html que será exibida no navegador. O outro caso é quando o cliente requisitar uma página escrevendo a requisição de forma errada, nesse caso ele é capaz de identificar e enviar o erro 400 de Bad Request

```
http_response = """HTTP/1.1 404 Not Found\r\n\r\n""" + contend

http_response = """HTTP/1.1 400 Bad Request\r\n\r\n""" + str(contend)
```

Por fim, tenho uma declaração da resposta do servidor, ele retorna o que foi solicitado pelo cliente e encerra a conexão e o socket do servidor.

```
client_connection.send(http_response.encode('utf-8'))

client_connection.close()

listen_socket.close()
```

LINKS

* Vídeo Youtube: https://youtu.be/DfsZggvXiTY

* Código no Github: https://github.com/FerCosta/servidor-http

PRINTS

Server.py > ...

```python
# Tarefa: Programação com sockets
# Fernanda Costa de Sousa

import socket

HOST = ''
PORT = 8081

listen_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

listen_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

listen_socket.bind((HOST, PORT))

listen_socket.listen(1)

print ('Serving HTTP on port %s ...' % PORT)

while True:
    client_connection, client_address = listen_socket.accept()

    request = client_connection.recv(1024).decode()

    request = request
    request = request.split(" ")
    print(request)
    if(request[0] == 'GET'):
        file = request[1].split(" ")
        print(file[0][1:])
        if(file[0] == "/"):
```

Figura 1

```
            index = open('index.html')
            contend = index.read()
            http_response = """HTTP/1.1 200 OK\r\n\r\n""" + str(contend)
            index.close()
        else:
            try:
                response = open(file[0][1:])
                contend = response.read()
                http_response = """HTTP/1.1 200 OK\r\n\r\n""" + str(contend)
                response.close()
            except:
                not_found = open("not_found.html")
                contend = not_found.read()
                http_response = """HTTP/1.1 404 Not Found\r\n\r\n""" + contend
                not_found.close()
    else:
        bad_request = open('bad_request.html')
        contend = bad_request.read()
        http_response = """HTTP/1.1 400 Bad Request\r\n\r\n""" + str(contend)
        bad_request.close()




    client_connection.send(http_response.encode('utf-8'))

    client_connection.close()

listen_socket.close()
```

Figura 2

index.html > ...

```
<!DOCTYPE html>

<html>
  <head>
    <title>Hello, World!</title>
  </head>

  <body>
    <p>Meu servidor funciona</p>
  </body>
</html>

```

Figura 3

not_found.html > ...

```
1  <!DOCTYPE html>
2
3  <html>
4    <head>
5      <title>Page not found!!</title>
6    </head>
7
8    <body>
9      <h1>404 Page Not Found</h1>
10     <p1>Sorry, the page you're looking for does not exist.</p1>
11   </body>
12 </html>
13
```

Figura 4