

# CK0249 - REDES DE COMPUTADORES I

## Unidade 2 - Programação em Sockets

Gabriel Camurça Fernandes de Sousa - 420549

Junho 20, 2021

## 1 Servidor

### 1.1 Código

```
1  from socket import *
2  import sys
3
4  def main():
5      server_host = ''
6      server_port = 3000
7
8      socket_instance = socket(AF_INET, SOCK_STREAM) #AF_INET = IPv4 SOCK_STREAM = IP
9      socket_instance.bind((server_host, server_port))
10     socket_instance.listen(1)
11     print("Servidor iniciado e pronto para estabelecer uma nova conexão")
12
13     while True:
14         conexao, endereco = socket_instance.accept()
15         print("Server conectado por", endereco)
16
17         while True:
18             content = conexao.recv(1024).decode()
19
20             if not content: break
21             else:
22                 try:
23                     file = open(f'./livros/{content}.txt', 'r')
24                     conexao.send(("O Livro " + content + " está disponível!").encode())
25                 except:
26                     conexao.send(("O Livro " + content + " não está disponível!").encode())
27
28         conexao.close()
29
30     print('Servidor pronto para estabelecer uma nova conexão')
31
32
33
34 if __name__ == '__main__':
35     main()
```

## 1.2 Explicação

Encontrado no arquivo ‘server.py’, o código em questão trata do que o servidor deve fazer uma vez recebida a entrada do cliente.

Em particular, foi implementado um sistema simples de checagem de livros: o usuário informa o nome do livro e o servidor responde se ele está disponível ou não.

Primeiramente, definimos o qual host e port serão utilizados pelo server, ou seja, a identificação dele para que o cliente saiba para onde fazer as requisições. No nosso caso, podemos utilizar a notação de uma string vazia para indicar que o endereço a ser utilizado é o da própria máquina; assim como especificamos um dos ports disponíveis da máquina: 3000.

Então, criamos um objeto socket passando como argumentos: `AF_INET`, para indicar que será utilizado o protocolo TCP; e `SOCK_STREAM`, indicando que será utilizado o protocolo IP

Fazendo a atribuição das informações de host e port ao socket, deixamos o servidor ‘escutando’ por uma possível conexão.

Ele então deve ficar disponível para que requisições sejam feitas pelo cliente de forma que mais de uma conexão possa ser feita sem fechar o servidor após a primeira terminar. Isso é feito através do primeiro loop infinito. Dentro dele, representando uma conexão estabelecida, temos outro loop infinito, que trata de continuar a conexão com o cliente e encerra quando o mesmo para de enviar mensagens.

É então que o tratamento da mensagem recebida ocorre: guardamos a mensagem e a analisamos para saber qual o nome do livro especificado pelo cliente. Se não houver mensagem para ser tratada a conexão é descontinuada; já se houver fazemos um processo de decodificação, visto que precisamos apenas do conteúdo que o cliente mandou, e checamos se é possível achar o livro em questão dentro da pasta que contém todos os livros.

Se for possível achar o arquivo do livro, uma mensagem de confirmação é mandada de volta para o cliente; caso contrário uma mensagem informando o cliente de que o livro não está disponível é mandada.

Então o servidor fecha a conexão estabelecida e fica à espera de uma nova.

## 2 Cliente

### 2.1 Código

```
1  from socket import *
2
3  def main():
4      server_host = 'localhost'
5      server_port = 3000
6
7      socket_instance = socket(AF_INET, SOCK_STREAM)
8      socket_instance.connect((server_host, server_port))
9
10     msg = input("Digite o nome do livro que deseja procurar: ")
11     msg_encoded = msg.encode()
12
13     socket_instance.send(msg_encoded)
14
15     content = socket_instance.recv(1024).decode()
16
17     print("Recebi:", content)
18
19     socket_instance.close()
20
21
22
23  if __name__ == '__main__':
24      main()
```

## 2.2 Explicação

Encontrado no arquivo 'client.py', o código em questão trata de como o cliente irá se comunicar com o servidor.

Em particular, o cliente se conecta com o servidor e especifica o nome de um livro qualquer, recebendo como resposta se o livro está disponível ou não.

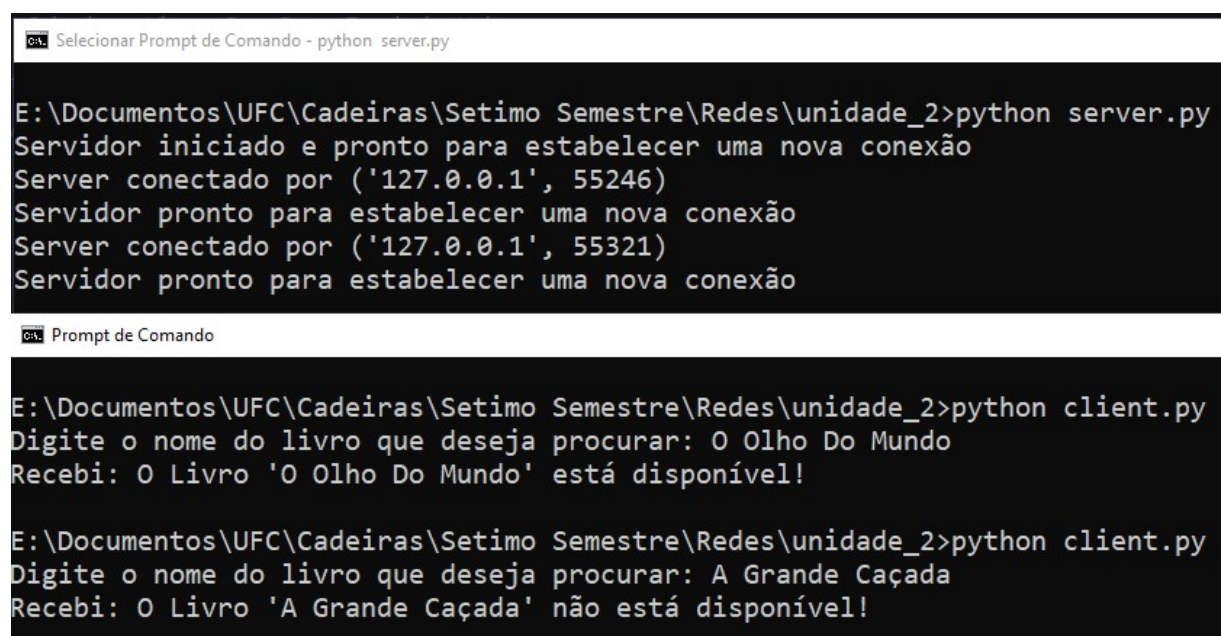
Primeiramente, definimos as informações do endereço do servidor a ser conectado; como será rodado localmente, 'localhost' é apropriado, juntamente com a porta 3000.

De forma análoga ao servidor, criamos um objeto socket passando como argumentos AF\_INET e SOCK\_STREAM, com o propósito de especificar a utilização dos protocolos TCP/IP.

As informações de host e port também são atribuídas ao socket e o cliente dispara uma tentativa de conexão com o servidor.

Uma vez estabelecida essa conexão, o nome do livro é especificado pelo cliente e depois de alguns momentos é retornado do servidor a resposta se o livro está disponível, encerrando então a conexão.

## 3 Execução dos Códigos



```
CA: Selecionar Prompt de Comando - python server.py

E:\Documentos\UFC\Cadeiras\Setimo Semestre\Redes\unidade_2>python server.py
Servidor iniciado e pronto para estabelecer uma nova conexão
Server conectado por ('127.0.0.1', 55246)
Servidor pronto para estabelecer uma nova conexão
Server conectado por ('127.0.0.1', 55321)
Servidor pronto para estabelecer uma nova conexão

CA: Prompt de Comando

E:\Documentos\UFC\Cadeiras\Setimo Semestre\Redes\unidade_2>python client.py
Digite o nome do livro que deseja procurar: O Olho Do Mundo
Recebi: O Livro 'O Olho Do Mundo' está disponível!

E:\Documentos\UFC\Cadeiras\Setimo Semestre\Redes\unidade_2>python client.py
Digite o nome do livro que deseja procurar: A Grande Caçada
Recebi: O Livro 'A Grande Caçada' não está disponível!
```