Atividade de programação 1: Servidor Web

Neste laboratório, você aprenderá os fundamentos da programação de soquetes para conexões TCP em Python: como criar um soquete, vinculá-lo a um endereço e porta específicos, bem como enviar e receber um pacote HTTP. Você também aprenderá alguns fundamentos do formato de cabeçalho HTTP.

Você desenvolverá um servidor da Web que lida com uma solicitação HTTP por vez. Seu servidor da Web deve aceitar e analisar a solicitação HTTP, obter o arquivo solicitado do sistema de arquivos do servidor, criar uma mensagem de resposta HTTP consistindo no arquivo solicitado precedido por linhas de cabeçalho e enviar a resposta diretamente ao cliente. Se o arquivo solicitado não estiver presente no servidor, o servidor deverá enviar uma mensagem HTTP "404 Not Found" de volta ao cliente.

Código

Abaixo você encontrará o esqueleto do código para o servidor Web. Você deve completar o esqueleto do código. Os lugares onde você precisa preencher o código são marcados com `#Fill in start` e `#Fill in end`. Cada local pode exigir uma ou mais linhas de código.

Executando o servidor

Coloque um arquivo HTML (por exemplo, HelloWorld.html) no mesmo diretório em que o servidor está. Execute o programa do servidor. Determine o endereço IP do host que está executando o servidor (por exemplo, 128.238.251.26). De outro host, abra um navegador e forneça a URL correspondente. Por exemplo:

http://128.238.251.26:6789/HelloWorld.html

'HelloWorld.html' é o nome do arquivo que você colocou no diretório do servidor. Observe também o uso do número da porta após os dois pontos. Você precisa substituir esse número de porta por qualquer porta que tenha usado no código do servidor. No exemplo acima, usamos o número da porta 6789. O navegador deve exibir o conteúdo de HelloWorld.html. Se você omitir ":6789", o navegador assumirá a porta 80 e você obterá a página da Web do servidor apenas se o seu servidor estiver escutando na porta 80.

Em seguida, tente obter um arquivo que não esteja presente no servidor. Você deve receber uma mensagem "404 Not Found".

O que entregar

Você entregará o código completo do servidor junto com as capturas de tela do navegador do cliente, verificando se realmente recebeu o conteúdo do arquivo HTML do servidor.

Esqueleto de código Python para o servidor da Web

```
#importar módulo de socket
from socket import *
import sys # Para encerrar o programa

serverSocket = socket(AF_INET, SOCK_STREAM)

#Prepare um soquete de servidor
#Fill in start
#Fill in end


while True:
	#Estabeleça a conexão
	print('Ready to serve...')
	connectionSocket, addr = #Fill in start	#Fill in end
	try:
		message = #Fill in start		#Fill in end
		filename = message.split()[1]
		f = open(filename[1:])
		outputdata = #Fill in start		#Fill in end

		#Enviar uma linha de cabeçalho HTTP para o soquete
		#Fill in start
		#Fill in end

		#Enviar o conteúdo do arquivo solicitado ao cliente
		for i in range(0, len(outputdata)):
			connectionSocket.send(outputdata[i].encode())
		connectionSocket.send("\r\n".encode())

		connectionSocket.close()
	except IOError:
		#Enviar mensagem de resposta para arquivo não encontrado
		#Fill in start
		#Fill in end

		#Fechar soquete do cliente
		#Fill in start
		#Fill in end

serverSocket.close()
sys.exit() #Terminate the program after sending the corresponding data
```

Exercícios Opcionais

1. Atualmente, o servidor da Web lida com apenas uma solicitação HTTP por vez. Implemente um servidor multiencadeado (*multithreaded*) capaz de atender várias solicitações simultaneamente. Usando threading, primeiro crie um thread principal no qual seu servidor modificado escuta clientes em uma porta fixa. Ao receber uma solicitação de conexão TCP de um cliente, ele configurará a conexão TCP em outra porta e atenderá a solicitação do cliente em um thread separado. Haverá uma conexão TCP separada em um thread separado para cada par de solicitação/resposta.
2. Em vez de usar um navegador, escreva seu próprio cliente HTTP para testar seu servidor. Seu cliente se conectará ao servidor usando uma conexão TCP, enviará uma solicitação HTTP ao servidor e exibirá a resposta do servidor como uma saída. Você pode assumir que a solicitação HTTP enviada é um método GET. O cliente deve receber argumentos de linha de comando especificando o endereço IP do servidor ou nome do host, a porta na qual o servidor está escutando e o caminho no qual o objeto solicitado é armazenado no servidor. A seguir está um formato de comando de entrada para executar o cliente.

```
client.py server_host server_port nome_do_arquivo
```