INSTITUTO INFNET

Aluno: CARLOS EDUARDO DA SILVA LOUVIZE

Professor: LUIZ PAULO MAIA

TP3

Desenvolvimento Python para Sistemas Operacionais e Redes

09/03/2021

Atividade 1 - O que é um processo cliente?

Um processo cliente é um processo que, ativamente, se conecta a um servidor e faz requisições a ele. É um processo ativo.

Atividade 2 - O que é um processo servidor?

Diferentemente do processo cliente, o processo servidor é passivo. Fica esperando a conexão e requisição de processos clientes e responde a eles.

Atividade 3 - A função socket() do módulo ‘socket’ de Python é responsável por criar um socket no processo tanto para protocolo TCP, quanto UDP. Como diferenciar se o socket a ser criado é TCP e UDP?

Quando há necessidade de maior confiabilidade na comunicação, deve-se utilizar o protocolo TCP. Esse protocolo é orientado a conexões, por isso existe uma fase dedicada a estabelecer a conexão entre as máquinas de forma a garantir a integridade e ordem correta da entrega dos dados.

O protocolo UDP não garante a entrega, nem a ordem dos dados. Em contrapartida é um protocolo mais rápido e deve ser utilizado quando a perda de alguns bytes é aceitável para garantir maior velocidade.

Atividade 4 - Para sockets TCP, responda:

1. Que sequência de chamadas de funções em Python deve ser realizada pelo cliente? (Não precisa especificar os parâmetros)

   a- Criar o socket (estrutura para controlar a conexão de rede).

   ```
   s = socket.socket (socket_family, socket_type)
   ```

   b- Conectar o socket local a um socket remoto (servidor).

   ```
   s.connect((endereco, porta))
   ```

   c- Comunicação com o lado servidor (send e recv).

   ```
   s.send(bytes)
   ```

   ```
   bytes = s.recv(quant)
   ```

   d- Fechar a conexão.

   ```
   s.close()
   ```

2. Que sequência de chamadas de funções em Python deve ser realizada pelo servidor? (Não precisa especificar os parâmetros)

   a- Criar o socket (estrutura para controlar a conexão de rede)

   ```
   s = socket.socket (socket_family, socket_type)
   ```

   b- Associar o socket a um ip e porta locais (bind).

   ```
   s.bind(address)
   ```

   c- Permitir que o socket aceite conexões (listen).

   ```
   s.listen()
   ```

   d- Aceitar a conexão de um cliente (accept).

   ```
   (endereco, porta) = s.accept()
   ```

   e- Comunicação com o lado cliente (send e recv).

   ```
   s.send(bytes)

   bytes = s.recv(quant)
   ```

   f- Fechar a comunicação.

   ```
   s.close()
   ```

3. Quais destas funções são bloqueantes, isto é, o processo fica esperando?

   A função accept é bloqueante. O servidor fica esperando a conexão para prosseguir com o código, assim que o cliente solicitar conexão e esta for estabelecida.

Atividade 5 - Para sockets UDP, responda:

1. Que sequência de chamadas de funções em Python deve ser realizada pelo cliente? (Não precisa especificar os parâmetros)

   a- Criar o socket (estrutura para controlar a conexão de rede).

   ```
   s = socket.socket (socket_family, socket_type)
   ```

   b- Comunicação com o lado servidor (send e recv).

   ```
   s.sendto(bytes, (endereco, porta))

   (bytes,(endereco, porta)) = s.recvfrom(quant)
   ```

   c- Fechar a conexão.

   ```
   s.close()
   ```

2. Que sequência de chamadas de funções em Python deve ser realizada pelo servidor? (Não precisa especificar os parâmetros)

   a- Criar o socket (estrutura para controlar a conexão de rede)

   ```
   s = socket.socket (socket_family, socket_type)
   ```

   b- Associar o socket a um ip e porta locais (bind).

   ```
   s.bind(address)
   ```

   c- Comunicação com o lado cliente (send e recv).

   ```
   s.sendto(bytes, (endereco, porta))

   (bytes,(endereco, porta)) = s.recvfrom(quant)
   ```

d- Fechar a comunicação.

```
s.close()
```

3. Quais destas funções são bloqueantes, isto é, o processo fica esperando?

   A função `s.recvfrom()` é bloqueante.

Atividade 6 - Para que serve o comando socket.bind()?

A função serve para associar o socket a um endereço (ip e porta) pelo qual será feita a comunicação.

Atividade 7 - Em sockets Python, como é representado um endereço de um processo remoto?

Por uma tupla que contém (endereço, porta).

A parte “endereço” da tupla pode conter uma URL, um IP; a parte porta contém a porta pela qual se deseja estabelecer a conexão, por exemplo, a porta 80 é utilizada para compartilhar documentos do protocolo HTTP.

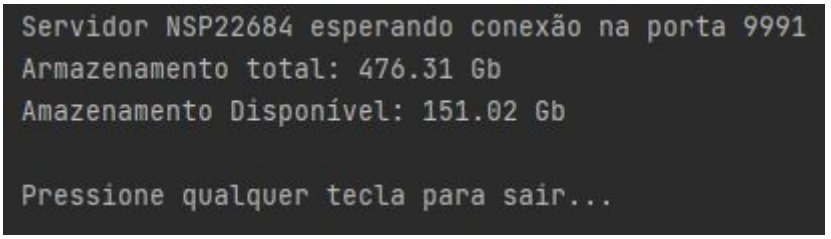
Exemplo de endereço chamado com a função “connect”:

```
# Cria uma INET (ou seja IPv4), SOCK_STREAM significa que a
conexão será feita utilizando o protocolo TCP
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Conecta ao servidor web utilizando a porta 80
s.connect(("www.python.org", 80))
```

Atividade 8 - Crie um programa cliente que:

1. Conecte-se a um servidor via UDP de mesmo IP e porta 9991.
2. Peça ao servidor que envie a quantidade total e disponível de armazenamento do disco principal.
3. Receba e exiba a informação.

Ver arquivo 08.py



Atividade 9 - Associado à questão anterior, crie um programa servidor que:

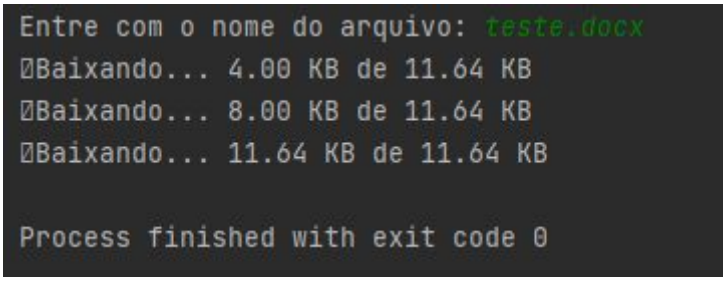
1. Espere conexões UDP de processos na porta 9991.
2. Aguarde indefinidamente conexão de clientes.
3. Sirva cada cliente com a informação da quantidade total e disponível de armazenamento do disco principal (diretório corrente que o processo servidor está executando).

Ver arquivo 09.py

Atividade 10 - Crie um programa cliente que:

1. Conecte-se a um servidor via TCP de mesmo IP e porta 8881.
2. Envie ao servidor o nome de um arquivo para que ele transmita este arquivo para o cliente.
3. Receba o tamanho do arquivo.
4. Se o tamanho for válido, receba o arquivo. Caso contrário, avise ao usuário que o arquivo não foi encontrado.

   Ver arquivo 10.py



Atividade 11 - Associado à questão anterior, crie um programa servidor que:

1. Espere conexões TCP de processos na porta 8881.
2. Aguarde indefinidamente conexão de clientes.
3. Receba a requisição do arquivo do cliente e envie o seu tamanho, caso o tenha encontrado. Em caso negativo, envie um valor inválido -1.
4. Envie o arquivo para o cliente, caso o encontre.

Ver arquivo 11.py