~~~~~~~~Cultura do Caracter~~~~~~~~~

>> Sockets em Python

#/usr/bin/python
#Sockets em Python
#
Por Alisson Menezes- GR1L0S3C0
Agradecimentos - Hederson Boechat - Guilherme Chagas - Talliny Dalla Nora - Walter Algusto
- André Zarlenga - Alex Moreno
#
culturadocaracter.com.br / botecounix.com.br
#

Sockets são usados para enviar dados através da rede, um exemplo seria enviar um arquivo pelo MSN, ou até mesmo as mensagens. Nesse tutorial, vou criar uma aplicação estilo messenger, cliente - servidor, e enviar as mensagens na rede fazendo o uso de Sockets.

Um grande erro que alguns programadores comentem, é achar que os Sockets são da linguagem de programação, quando na realidade, já são do próprio Sistema Operacional, precisamos somente saber como usa-los.

Como saber um pouco mais sobre Socket em qualquer Linux, é só usar o comando:

$ man socket

Essa é a documentação que te dá uma visão geral sobre o seu uso e suas funções, depois é só adaptar conforme a linguagem de programação que estiver usando.

Aplicação do lado Servidor

Primeiro, vamos criar o servidor que vai receber as mensagens da nossa aplicação Client.

##########=====serv_tuto.py=====##########
#!/user/bin/python
#
Coded by: Alisson Menezes [kernelcrash]
#
servidor que recebe mensagens de aplicação client parecido com o netsend
#
#

```
import socket

host = ''
port = 7000

addr = (host, port)
serv_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serv_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
serv_socket.bind(addr)
serv_socket.listen(10)
```
~~~~~~~~

```
print 'aguardando conexao'
con, cliente = serv_socket.accept()
print 'conectado'
print "aguardando mensagem"
recebe = con.recv(1024)

print "mensagem recebida: "+ recebe
serv_socket.close()
##############=====EOF======###################
```

Explicando um pouco as linhas acima:

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

- Nesta, criamos o nosso mecanismo de Socket para receber a conexão, onde na função passamos 2 argumentos, AF_INET que declara a família do protocolo; se fosse um envio via Bluetooth por exemplo, seria: AF_BLUETOOTH, SOCKET_STREAM, indica que será TCP/IP.

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

- Essa linha serve para zerar o TIME_WAIT do Socket, por exemplo, se o programa estiver aguardando uma conexão e você der CTRL+C para interromper, o programa fecha porém o Socket continua na escuta e se você for usar a mesma porta receberá a seguinte mensagem:
socket.error: [Errno 98] Address already in use

serv_socket.bind(addr)

- Esta linha define para qual IP e porta o servidor deve aguardar a conexão, que no nosso caso é qualquer IP, por isso o Host é ".

serv_socket.listen(10)

- Define o limite de conexões.

con, cliente = serv_socket.accept()

- Deixa o Servidor na escuta aguardando as conexões, feita a conexão vem o seguinte comando:

recebe = con.recv(1024)

Que aguarda um dado enviado pela rede de até 1024 Bytes, a função 'recv' possui somente 1 argumento que é o tamanho do Buffer.

Aplicação do lado Cliente

Agora, vamos criar a aplicação cliente para testar o nosso Servidor:

```
############## client_tuto.py#################
#!/usr/bin/python
#Coded by: Alisson Menezes [kernelcrash]
```

```
#

import socket

ip = raw_input('digite o ip de conexao: ')
port = 7000
addr = ((ip,port))
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client_socket.connect(addr)

mensagem = raw_input("digite uma mensagem para enviar ao servidor")
client_socket.send(mensagem)
print 'mensagem enviada'
client_socket.close()

############=----EOF----###########
```

A maioria das linhas do nosso programa Client já foi descrita no Servidor, pois são apenas alguns ajustes para se fazer o uso do Socket, neste segundo programa, a única diferença são as linhas:

client_socket.connect(addr)

- Que faz a conexão no nosso servidor.

client_socket.send(mensagem)

- Que faz o envio do dado para servidor.

client_socket.close()

- Serve para fechar a conexão entre os dois aplicativos.

Bem, acho que um resumão sobre Socket em Python.

É isso aí, em breve faço um novo tutorial fazendo o uso de Threads.