Universidade Federal de Uberlândia
Engenharia de Controle e Automação / Engenharia Mecatrônica
Sistemas Embarcados II / Sistemas Digitais para Mecatrônica
Prof. Éder Alves de Moura
Semana 06 – Comunicação Interprocesso com Sockets

Aula Pratica :

```
fischmannn@fischmannn-VirtualBox:~$ sudo pip3 install beautifulsoup4
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.10.0-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 2.0 MB/s
Collecting soupsieve>1.2
  Downloading soupsieve-2.3.1-py3-none-any.whl (37 kB)
Installing collected packages: soupsieve, beautifulsoup4
Successfully installed beautifulsoup4-4.10.0 soupsieve-2.3.1
fischmannn@fischmannn-VirtualBox:~$ sudo pip3 install requests
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.22.
0)
fischmannn@fischmannn-VirtualBox:~$ sudo pip3 install html5lib
Collecting html5lib
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     |████████████████████████████████| 112 kB 3.3 MB/s
Requirement already satisfied: six>=1.9 in /usr/lib/python3/dist-packages (from
html5lib) (1.14.0)
Collecting webencodings
  Downloading webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
Installing collected packages: webencodings, html5lib
Successfully installed html5lib-1.1 webencodings-0.5.1
fischmannn@fischmannn-VirtualBox:~$ S
```

```
============== RESTART: /home/fischmannn/Desktop/semana6/server.py ==========
Esperando por nova conexao!!
recebido:.b'Messagem enviada pelo cliente'
reenviando mensagem para o cliente:  ('127.0.0.1', 50020)
recebido:.b'Messagem enviada pelo cliente'
reenviando mensagem para o cliente:  ('127.0.0.1', 50020)
recebido:.b'Messagem enviada pelo cliente'
reenviando mensagem para o cliente:  ('127.0.0.1', 50020)
recebido:.b'Messagem enviada pelo cliente'
reenviando mensagem para o cliente:  ('127.0.0.1', 50020)
recebido:.b'Messagem enviada pelo cliente'
reenviando mensagem para o cliente:  ('127.0.0.1', 50020)
recebido:.b''
nao foi recebido nenhum dado de:  ('127.0.0.1', 50020)
Esperando por nova conexao!!
```

```
============== RESTART: /home/fischmannn/Desktop/semana6/client.py =========
Conectando à - localhost:30000
Pressione enter para continuar
1 b'Messagem enviada pelo cliente'
b'Messagem enviada pelo cliente'
Pressione enter para continuar
2 b'Messagem enviada pelo cliente'
b'Messagem enviada pelo cliente'
Pressione enter para continuar
3 b'Messagem enviada pelo cliente'
b'Messagem enviada pelo cliente'
Pressione enter para continuar
4 b'Messagem enviada pelo cliente'
b'Messagem enviada pelo cliente'
Pressione enter para continuar
5 b'Messagem enviada pelo cliente'
b'Messagem enviada pelo cliente'
finalizando a conexão
```

3 – Implemente e teste códigos disponíveis em:

- Transfer file over TCP/UDP:
https://chuanjin.me/2016/08/03/transfer-file/

e responda:

a) com comentários ao longo do código, explique o cada linha está do código está fazendo

semana 6 > tcp_sender.py > ...

```python
import socket
import sys

#define as portas e IPs
TCP_IP = "127.0.0.1"
FILE_PORT = 5005
DATA_PORT = 5006
buf = 1024
file_name = sys.argv[1]


try:
    #tenta establecar a conecção om a porta do IP definido por TCP
    #para mandar o documento
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((TCP_IP, FILE_PORT))
    sock.send(file_name)
    sock.close()

    print "Sending %s ..." % file_name

    #abre o documento
    f = open(file_name, "rb")
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((TCP_IP, DATA_PORT))
    data = f.read()
    #manda o conteudo do documento no endereco definido
    sock.send(data)

finally:
    #fecha o socket
    sock.close()
    f.close()
```

semana 6 > tcp_receiver.py > ...

```python
import socket


#define o IP e as portas do documento e dos dados
TCP_IP = "127.0.0.1"
FILE_PORT = 5005
DATA_PORT = 5006
timeout = 3
buf = 1024

#abre a conecção com o tcp_sender
sock_f = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock_f.bind((TCP_IP, FILE_PORT))
sock_f.listen(1)

sock_d = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock_d.bind((TCP_IP, DATA_PORT))
sock_d.listen(1)


while True:
    conn, addr = sock_f.accept()
    data = conn.recv(buf)
    if data:
        print "File name:", data
        file_name = data.strip()

    f = open(file_name, 'wb')
    #abre o documento recebido

    conn, addr = sock_d.accept()
    while True:
        #receve cada dado no documento recebido
        data = conn.recv(buf)
        if not data:
            break
        f.write(data)

    print "%s Finish!" % file_name
    f.close()
```

semana 6 > udp_sender.py > ...

```python
import socket
import time
import sys

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
buf = 1024
file_name = sys.argv[1]

#manda os dados do socket para  o endereço definido
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(file_name, (UDP_IP, UDP_PORT))
print "Sending %s ..." % file_name

#abre o documento e le os dados dentro dele
f = open(file_name, "r")
data = f.read(buf)
while(data):
    #se o dado foi enviado
    if(sock.sendto(data, (UDP_IP, UDP_PORT))):
        #le o bufer dos dados
        data = f.read(buf)
        time.sleep(0.02) # Give receiver a bit time to save

sock.close()
f.close()
```

semana 6 > udp_receiver.py > ...

```python
import socket
import select

UDP_IP = "127.0.0.1"
IN_PORT = 5005
timeout = 3


sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, IN_PORT))

while True:
    data, addr = sock.recvfrom(1024)
    #receve os dados e o edereco de udp_sender
    if data:
        #se recebe dados, mostra os dados
        print "File name:", data
        #concatena os dados no documento
        file_name = data.strip()
    #abre o documento
    f = open(file_name, 'wb')

    while True:
        ready = select.select([sock], [], [], timeout)
        if ready[0]:
            data, addr = sock.recvfrom(1024)
            f.write(data)
            #continua concatenando dados no documento se recebe mais dados
        else:
            print "%s Finish!" % file_name
            f.close()
            #se não termina o programa
            break
```

4 – Implemente o programa de chat apresentado no seguinte vídeo:

- Tutorial de Socket com Python - Guia Completo de Soquetes:
https://www.youtube.com/watch?v=VhhNlWdLPzA

Comente o código, explicando sua funcionalidade.

Server.py

```python
import socket
import threading
import time

#define o ip do servidor coo endereco IP do HOST
SERVER_IP = socket.gethostbyname(socket.gethostname())
```

```
PORT = 5050
ADDR = (SERVER_IP, PORT)
#define o formato de comunicação como utf-8
FORMATO = 'utf-8'

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

conexoes = []
mensagens = []

def enviar_mensagem_individual(conexao):
    #Essa função manda a mensagem i da lista mensagens para uma conexao
    print(f"[ENVIANDO] Enviando mensagens para {conexao['addr']}")
    for i in range(conexao['last'], len(mensagens)):
        mensagem_de_envio = "msg=" + mensagens[i]
        conexao['conn'].send(mensagem_de_envio.encode())
        conexao['last'] = i + 1
        time.sleep(0.2)

def enviar_mensagem_todos():
    #Essa função envia todas as mensagens da lista mensagens para
    #as conexoes da lista conexao
    global conexoes
    for conexao in conexoes:
        enviar_mensagem_individual(conexao)

"""
1 vez que o cliente entrar, vai mandar o nome:
nome=.....
E as mensagens vem:
msg=
"""

def handle_clientes(conn, addr):
    #Essa função recebe o endereço, a conexão e o nome de um cliente
    #e envia uma mensagem individual para ele
    #ou envia todas as mensagens para os clientes
    print(f"[NOVA CONEXAO] Um novo usuario se conectou pelo endereço {addr}")
    global conexoes
    global mensagens
    nome = False

    while(True):
        msg = conn.recv(1024).decode(FORMATO)
        if(msg):
            if(msg.startswith("nome=")):
                mensagem_separada = msg.split("=")
                nome = mensagem_separada[1]
```

```
                mapa_da_conexao = {
                    "conn": conn,
                    "addr": addr,
                    "nome": nome,
                    "last": 0
                }
                conexoes.append(mapa_da_conexao)
                enviar_mensagem_individual(mapa_da_conexao)
            elif(msg.startswith("msg=")):
                mensagem_separada = msg.split("=")
                mensagem = nome + "=" + mensagem_separada[1]
                mensagens.append(mensagem)
                enviar_mensagem_todos()



def start():
    #abre o server e establece a conexão como thread
    print("[INICIANDO] Iniciando Socket")
    server.listen()
    while(True):
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_clientes, args=(conn, addr))
        thread.start()

start()
```

Client.py

```
import socket
import threading
import time

PORT = 5050
FORMATO = 'utf-8'
SERVER = "192.168.0.109"
ADDR = (SERVER, PORT)

#abre o socket e se conecta no endereço definido
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)

def handle_mensagens():
    #esssa funcao trata a mensagem enviada separando ela pelo '='
    while(True):
        msg = client.recv(1024).decode()
        mensagem_splitada = msg.split("=")
```

```
        print(mensagem_splitada[1] + ": " + mensagem_splitada[2])

def enviar(mensagem):
    #enssa função envia a mensagem no formato utf-8
    client.send(mensagem.encode(FORMATO))

def enviar_mensagem():
    #essa função pede ao utilizador de escrever a mensagem a ser enviada
    mensagem = input()
    enviar("msg=" + mensagem)

def enviar_nome():
    #essa função envia o nome digitado pelo utilizador
    nome = input('Digite seu nome: ')
    enviar("nome=" + nome)

def iniciar_envio():
    #o utilizador digita o nome e a mensagem e são enviados
    enviar_nome()
    enviar_mensagem()

def iniciar():
    #O thread 1 mostra as mensagens enviadas
    thread1 = threading.Thread(target=handle_mensagens)
    #O thread 2 pede o nome e a mensagem do utilizados para mandar os dois
    thread2 = threading.Thread(target=iniciar_envio)
    thread1.start()
    thread2.start()

iniciar()
```