PDI - TP 1 - C - Cliente Servidor utilizando Sockets en Python. (Entrega 13/05/2024)

Para realizar programas Cliente Servidor con Python utlizamos la librería o paquete socket.py
(https://github.com/python/cpython/blob/3.10/Lib/socket.py)). Esta librería es una transcripción sencilla de la llamada al sistema sockets de BSD Unixal estilo orientado a objetos de Python:. La función socket() devuelve a socket object métodos que implementan las diversas llamadas al sistema de socket. Los tipos de parámetros tienen un nivel algo más alto que en la interfaz C, como con read() y write() en el uso de los archivos Python, la asignación del buffer es automática y la longitud del buffer está implícita en las operaciones de envío.

Para detalles de la implementación de socket.py ver;

Para la parte C del trabajo práctico usar como ejemplos de base los siguientes códigos (la explicación de la implementación está detallada dentro del mismo código)

y escribir una aplicación cliente servidor que muestre las direcciones y puertos de todos los clientes conectados del lado del servidor y devuelva a cada cliente el día y hora de conexión, y el tiempo que estuvo (o está conectado).

Realizar la misma aplicación tanto para C-S con Concurrencia Aparente (Select) como C-S Concurrente.

De ser necesario agregar tiempo de espera, loop, sleep, con contadores para demorar los procesos.

Implementar una limpieza de recuersos al salir del los programas (agregar opción de pregunta al usuario para cerrar los clientes).

Para crear un socket (stream) en Python: socket.socket(family=AF_INET, type=SOCK_STREAM, proto=0, fileno=None)

Los parámetros son los mismos que se usan en C

```
import socket
Creo socket IPv4
sock_fd = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
if sock_fd == -1:

Administro el error

Ejemplo de cliente sencillo

from socket import socket as Socket
from socket import AF_INET, SOCK_STREAM
SERVIDOR = 'a.b.c.d'  # IP
```

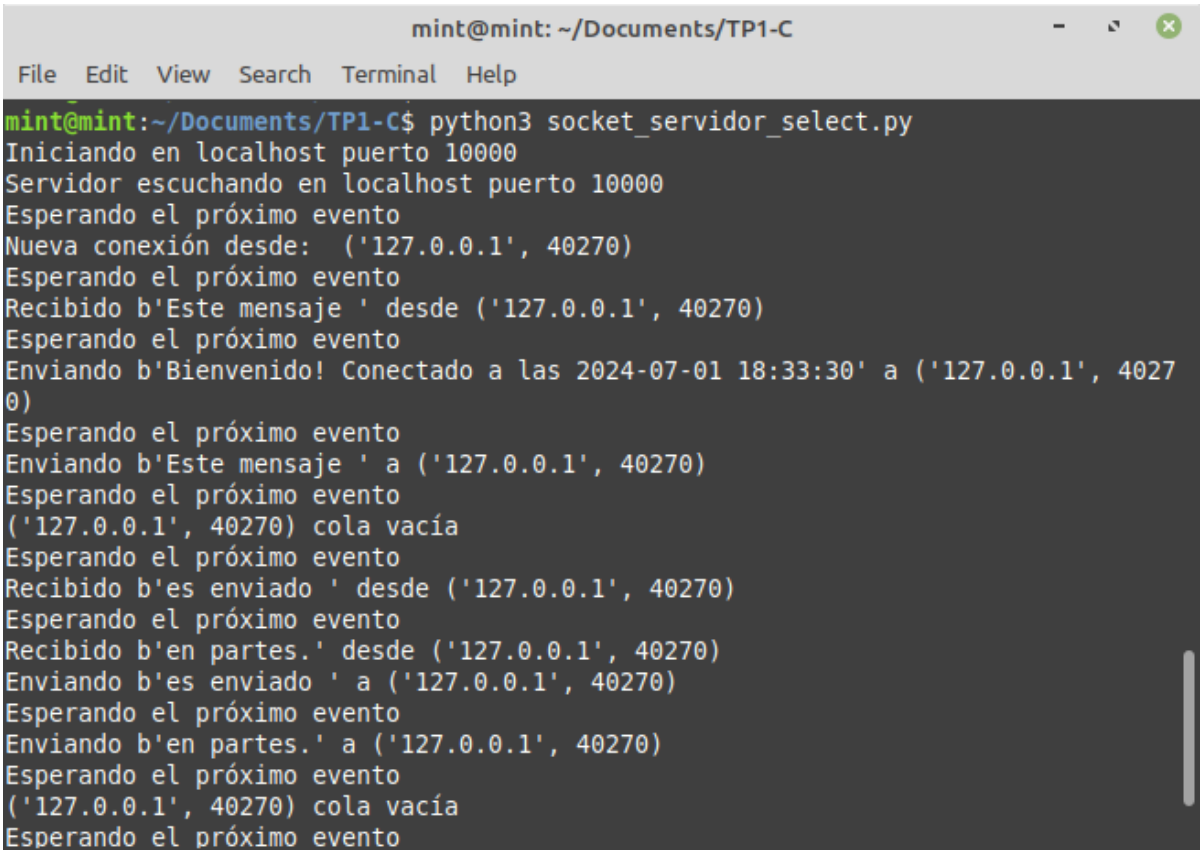
```
NROPUERTO = 41267         # puerto
BUFFER = 80               # tamaño del buffer

DIRECCION_SERVIDOR = (SERVIDOR, NROPUERTO)
CLIENTE = Socket(AF_INET, SOCK_STREAM)
try:
      CLIENTE.connect(SERVER_ADDRESS)
      print('cliente conectado')
      DATOS = input('Mensaje : ')
      CLIENTE.send(DATOS.encode())
except OSError:
      print('connection failed')
CLIENT.close()
```

TP1 – C – Soluciones

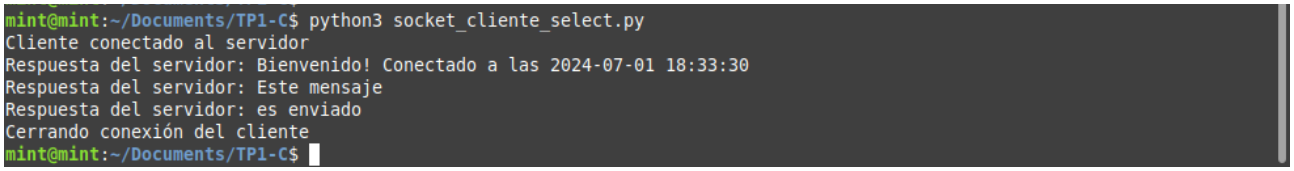
Para los programas socket servidor con SELECT, se manejan multiples conexiones de clientes entrantes. Lo que hace el siguiente programa es enviar la fecha y hora de conexion al cliente y calcula el tiempo de esa conexion.

Utilizo el puerto 10000.
Guarda cada cliente conectado en ‘clientes_info’.
El mensaje inicial que envia el cliente, recibe la respuesta con fecha y hora de conexion.
Finalmente se cierra el socket cliente al terminar la cola de mensajes, se elimina de la lista de clientes y se limpian los recursos.

Salida Servidor



Salida Cliente



Codigo servidor

```
import socket
import select
import queue
import sys
import datetime
import time

# Creando un socket TCP/IP
servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
servidor.setblocking(0)

# Hago Bind del socket al puerto
dir_servidor = ('localhost', 10000)
print('Iniciando en {} puerto {}'.format(*dir_servidor), file=sys.stderr)
servidor.bind(dir_servidor)

# Escucho conexiones entrantes
servidor.listen(5)

# Sockets que espero leer
entradas = [servidor]

# Sockets que espero enviar
salidas = []

# Cola de mensajes salientes
cola_mensajes = {}

# Diccionario para mantener información de clientes conectados
clientes_info = {}

print('Servidor escuchando en {} puerto {}'.format(*dir_servidor), file=sys.stderr)

# Función para obtener la fecha y hora actual
def obtener_fecha_hora():
    now = datetime.datetime.now()
    return now.strftime("%Y-%m-%d %H:%M:%S")

# Función para manejar la desconexión de un cliente
```

```
def desconectar_cliente(cliente_socket):
    if cliente_socket in clientes_info:
        # Calcular tiempo total de conexión
        if cliente_socket in clientes_info:
            tiempo_conexion = time.time() - clientes_info[cliente_socket]["connect_time"]
            tiempo_conexion_str = f"{tiempo_conexion:.2f} segundos"

            print(f'Cerrando conexión desde {clientes_info[cliente_socket]["address"]}. Tiempo
conectado: {tiempo_conexion_str}', file=sys.stderr)

            # Enviar tiempo de conexión al cliente antes de desconectarlo
            mensaje_desconexion = f"Desconectado. Tiempo total conectado: {tiempo_conexion_str}"
            cola_mensajes[cliente_socket].put(mensaje_desconexion.encode())

        # Dejar de escuchar en la conexión
        if cliente_socket in salidas:
            salidas.remove(cliente_socket)
        entradas.remove(cliente_socket)
        cliente_socket.close()

        # Remover mensaje de la cola si existe
        if cliente_socket in cola_mensajes:
            del cola_mensajes[cliente_socket]

        # Remover información del cliente si existe
        if cliente_socket in clientes_info:
            del clientes_info[cliente_socket]

# Función para enviar mensaje inicial al cliente y calcular tiempo conectado
def enviar_mensaje_inicial(cliente_socket):
    fecha_hora = obtener_fecha_hora()
    mensaje_inicial = f"Bienvenido! Conectado a las {fecha_hora}"
    cola_mensajes[cliente_socket].put(mensaje_inicial.encode())

    # Guardar el tiempo de conexión
    clientes_info[cliente_socket] = {
        "address": cliente_socket.getpeername(),
        "connect_time": time.time()
    }

# Función para manejar eventos de selección
while entradas:
    # Espero a que al menos uno de los sockets esté listo para ser procesado
    print('Esperando el próximo evento', file=sys.stderr)
    try:
        readable, writable, exceptional = select.select(entradas, salidas, entradas)
    except select.error as e:
        print('Error de select:', e, file=sys.stderr)
        break

    if not (readable or writable or exceptional):
        print('Tiempo excedido....', file=sys.stderr)
```

```
    continue

# Manejo de sockets listos para leer
for s in readable:
    if s is servidor:
        # Un socket "leíble" está listo para aceptar conexiones
        cliente_socket, dir_cliente = s.accept()
        print('Nueva conexión desde: ', dir_cliente, file=sys.stderr)
        cliente_socket.setblocking(0)
        entradas.append(cliente_socket)

        # Inicializar cola de mensajes para el cliente
        cola_mensajes[cliente_socket] = queue.Queue()

        # Enviar mensaje inicial al cliente y calcular tiempo conectado
        enviar_mensaje_inicial(cliente_socket)

    else:
        # Socket cliente tiene datos para leer
        data = s.recv(1024)
        if data:
            # Datos recibidos, encolar para enviar de vuelta al cliente
            print('Recibido {!r} desde {}'.format(data, s.getpeername()), file=sys.stderr)
            cola_mensajes[s].put(data)

            # Agregar el socket a salidas si no está presente
            if s not in salidas:
                salidas.append(s)
        else:
            # Datos vacíos, cliente desconectado
            desconectar_cliente(s)

# Manejo de sockets listos para escribir
for s in writable:
    if s in cola_mensajes:
        try:
            next_msg = cola_mensajes[s].get_nowait()
        except queue.Empty:
            print('{} cola vacía'.format(s.getpeername()), file=sys.stderr)
            salidas.remove(s)
        else:
            try:
                print('Enviando {!r} a {}'.format(next_msg, s.getpeername()), file=sys.stderr)
                s.send(next_msg)
            except ConnectionResetError:
                print('Error al enviar datos a', s.getpeername(), file=sys.stderr)
                desconectar_cliente(s)

# Manejo de condiciones excepcionales
for s in exceptional:
    print('Excepción en', s.getpeername(), file=sys.stderr)
    # Desconectar cliente
```

```
        desconectar_cliente(s)
```

Codigo cliente

```
import socket

# Creo socket IPv4
cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Conectar el socket al puerto en el cual el servidor está escuchando
dir_servidor = ('localhost', 10000)
cliente.connect(dir_servidor)

print('Cliente conectado al servidor')

# Ejemplos de mensajes a enviar
mensajes = [
    'Este mensaje ',
    'es enviado ',
    'en partes.',
]

try:
    for mensaje in mensajes:
        # Envío mensajes al servidor
        cliente.sendall(mensaje.encode())

        # Espero la respuesta del servidor
        data = cliente.recv(1024)
        print('Respuesta del servidor:', data.decode())

finally:
    print('Cerrando conexión del cliente')
    cliente.close()
```

*__se implemento la libreria proporcionada socket.py especificada__