Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: Patricio Villalon
Actualizado por Luis Pereira
Sergio Villena
2023-2

Practico Nº2 Comunicación Serial y Socket

Este código es un servidor de chat que permite que dos usuarios se conecten entre sí para chatear. El servidor escucha las conexiones entrantes y las maneja usando hilos. El usuario también puede enviar un mensaje a todos los usuarios conectados. El servidor también registra todos los eventos y los muestra en una ventana de registro.

Aclaración: es importante desactivar el firewall y conocer la ip del servidor para poder conectarlo

Desarrollo

1 Agregue las siguientes bibliotecas:

```
import socket
import threading
import tkinter as tk
```

2 Crearemos las variables necesarias para poder conectarnos como servidor

```
HOST = ''  # Significa que escucha en todas las interfaces de red
PORT = 8888  # Puerto para escuchar
MAX_CONNECTIONS = 8  # Número máximo de conexiones simultáneas
```

3 Crearemos nuestra ventana de tkinter

4

```
#Ventana
ventana = tk.Tk()
ventana.title("Servidor")
#Label
EstadoLabel = tk.Label(ventana, text='Servidor detenido')
EstadoLabel.pack()

#Text
Log_Text = tk.Text(ventana, height=10,width=50)
Log_Text.pack()

#Botones
Start_Button = tk.Button(ventana, text='Iniciar', command=iniciar_Servidor)
Start_Button.pack()

Stop_Button = tk.Button(ventana, text='Detener', command=detener_Servidor, state=tk.DISABLED)
Stop_Button.pack()

#Text
mensaje_Text = tk.Text(ventana, height=3, width=50)
mensaje_Text.pack()

#Boton
Enviar_Button = tk.Button(ventana, text='Enviar',command=enviar_Mensaje,state=tk.DISABLED)
Enviar_Button.pack()


ventana.mainloop()
```

5 Crearemos la función que inicia el servidor: Esta función inicia el servidor. Crea un hilo para ejecutar la función correr_Servidor y luego habilita los botones necesarios para que el usuario pueda enviar y detener el servidor.

```
def iniciar_Servidor():
    Servidor_Thread = threading.Thread(target=correr_Servidor)
    Servidor_Thread.start()

    Start_Button.config(state=tk.DISABLED)
    Stop_Button.config(state=tk.NORMAL)
    mensaje_Text.config(state=tk.NORMAL)
    Enviar_Button.config(state=tk.NORMAL)

    EstadoLabel.config(text='Servidor corriendo')
```

6. Crearemos la función para detener el servidor: Esta función detiene el servidor. Cierra el socket y deshabilita los botones necesarios para que el usuario no pueda enviar o detener el servidor.

```
def detener_Servidor():
    Server_Socket.close()

    Start_Button.config(state=tk.NORMAL)
    Stop_Button.config(state=tk.DISABLED)
    mensaje_Text.config(state=tk.DISABLED)
    Enviar_Button.config(state=tk.DISABLED)
    EstadoLabel.config(text='Servidor Detenido')
```

7. Crearemos la función log: Esta función registra los eventos en una ventana de registro.

```
def log(mensaje):
    Log_Text.insert(tk.END, mensaje + '\n')
    Log_Text.see(tk.END)
```

8. Crearemos la función para Manejar_conexion: Esta función maneja la conexión con un cliente. Recibe los mensajes del cliente, los procesa y envía la respuesta.

```python
def Manejar_conexion(conn, addr, name):
    log(f'{name} conectado por {addr}')

    while True:
        #Recibimos los datos del cliente
        data = conn.recv(1024)
        mensaje = data.decode().strip()

        if not mensaje:
            #Cliente termina de enviar el mensaje
            break

        log(f'Datos recibidos de {name}: {mensaje}')

        #Procesamos el mensaje del cliente
        respuesta = f'Recibido: {mensaje}'.encode()

        #Enviamos la respuesta al cliente
        conn.sendall(respuesta)

    #cerrar la conexion
    conn.close()
    log(f'Conexión cerrada con {name}')
```

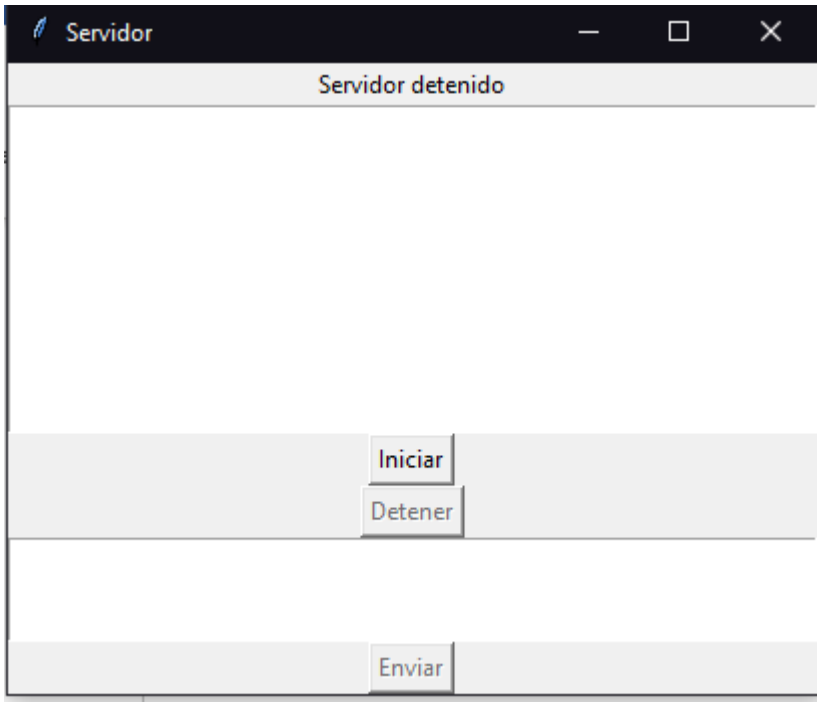
9. Crearemos la función correr_servidor: : Esta función inicia el socket, escucha las conexiones entrantes, acepta las conexiones y crea hilos para cada conexión.

```python
def correr_Servidor():
    # Crear un objeto de socket TCP
    global Server_Socket
    Server_Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Vincular el socket a una dirección y puerto específicos
    Server_Socket.bind((HOST, PORT))
    log(f'Servidor corriendo en el puerto {PORT}')

    # Escuchar en el socket para conexiones entrantes
    Server_Socket.listen(MAX_CONNECTIONS)

    #Guardar informacion de los clientes
    global connections
    connections = []
    names = []

    #Ciclo para manjear las conexciones
    while True:
        conn, addr = Server_Socket.accept()
        connections.append(conn)

        #Recibimos el nombre del cliente
        data = conn.recv(1024)
        name = data.decode().strip()
        names.append(name)

        #Creamos un hilo para las conexiones
        t = threading.Thread(target=Manejar_conexion, args=(conn, addr, name))
        t.start()
```

10. Crearemos la función enviar_Mensaje: Esta función toma el mensaje escrito por el usuario, lo envía a todos los usuarios conectados y lo registra en la ventana de registro.

```
def enviar_Mensaje():
    mensaje = mensaje_Text.get(1.0, tk.END).strip()
    mensaje_Text.delete(1.0,tk.END)
    log(f'Mensaje enviado a los clientes: {mensaje}')
    for conn in connections:
        conn.sendall(mensaje.encode())
    pass
```

11. Si lo ejecutamos se vería de esta forma cuando este detenido.



12. Si lo iniciamos se vería de esta forma.