PRACTICA 1

REDES DE ORDENADORES

José María Fernández Gómez

Practica Base:

1. Lectura del documento “MODBUS”.

2. Instalar Python.

En mi qeuipo python ya estaba instalado previamente, ya que he desarrollado otras aplicaciones en este lenguaje, como estoy empleando Arch-Linux el comando de instalación desde terminal es $pacman -S python python3 pip
De esta manera instalo python base, python3 y el gestor de paquetes pip

3. Instalar Librería pyModbusTCP. Vea el siguiente recurso:

Se ha instalado con el comando pip3 install pyModbudTCP

Codigo del cliente comentado:

```
#!/usr/bin/env python3

#(1) Importamos la libreria ModbusCliente
from pyModbusTCP.client import ModbusClient

client = ModbusClient(host="127.0.0.1", port=23456) # Se crea el cliente en la direccion ip
                                                    #local ya que se  correrá el servidor en
                                                    #la misma máquina bajo la dirección ip
                                                    #local en el puerto 23456


client.open() #Se abre la conexió del cliente hacia el srvidor con los parámetros definidos a
              #la hora de crear el objeto cliente


client.read_holding_registers(0)#Lee el valor del registro de direccion 0


client.write_single_register(1,55)#Se sobreescribe el registro de la dirección 1 con el valor 55


client.write_multiple_registers(1,[1, 2, 3])#Se sobreescriben tres registors a partir de la
                                            #direccion numero 1 (incluida) por lo que el registro
                                            #1 tomará el valor 1, el dos tomará el valor 2 y el
                                            #tres se escribirá con un 3
```

Codigo del servidor comentado:

```
#!/usr/bin/env python3

#Importamos librerias necesarias
from pyModbusTCP.server import ModbusServer, DataBank
from time import sleep
from random import uniform

server = ModbusServer("127.0.0.1", 23456, no_block=True) #Se crea un objeto servidor en
la                                                        #direccion ip local ya que el cliente
                                                          #se va acorrer en la misma
                                                          #maquina, se presta el servicio en
                                                          #el puerto 23456

try:
 print("Start server…") #Notificacion para ver por consola el estado del servidor
 server.start() #Se inicia en el puerto 23456 el servicio servidor
 print("Server is on line") #Notificacion para ver por consola el estado del servidor
 state = [0] #Array con valores iniciales para algunos registros, se usará para comprobar si
             #algun registro ha cambiado
 while True: #Bucle infinito para poner el servidor en escucha
 DataBank.set_words(0,[int(uniform(0,100))]) #Se generan numeros aleatorios entre 0 y 100
                                             #para rellenar los registros del servidor
 if state != DataBank.get_words(1): #Comprobamos si el valor del registro ha cambiado
 state = DataBank.get_words(1)  #Si el valor ha cambiado modificamos el valor de state para
                                #indicarle su nuevo valor
 print("value register 1 has changed to " + str(state)) #Notificamos por pantalla que el valor
                                                        #del registro ha sido modificado por un
                                                        #cliente
 sleep(0.5) #Paramos medio segundo la ejecución para poder ver por pantalla
except: #Gesinonamos cualquier error
 print("Shutdown server…") #Notificamos que el servidor ha caido
 server.stop() #Paramos el servicio
 print("Server is offline")
```

Practica Adicional:

Se incluyen los códigos ordenads por carpetas en función del ejercicio en el comprimido entregable, a continuación se comentará lo más relevante de cada apartado ya que son bastante similares entre sí.

En todos los apartados se ha incluido el servidor original proporcionado en la práctica y uno modificado para poder entender a la par que se ejecutan los clientes que está sucediendo en todo momento. Todos los clientes son funcionales tanto con el servidor original como con el modificado.

6.1.- Realice un programa cliente que escriba 5 registros sobre el servidor desde la dirección 5 y el resultado lo muestre el cliente con un mensaje indicativo de la acción realizada. Para ello el cliente debe leer los 5 registros desde la dirección 5 en el servidor..

```
#!/usr/bin/env python3

from pyModbusTCP.client import ModbusClient

client = ModbusClient(host="192.168.1.146", port=23456)

client.open()
#Lee los valores iniciales de los 5 registros desde la posicion 5

print("Initial register values :" + str(client.read_input_registers(5, reg_nb=5)))

#Escribe 4 registors desde la direccion 5 hasta la 25 con valores 3,2,1,3,4 respectivamente

client.write_multiple_registers(5,[3,2,1,3,4])

#Lee los valores recien escritos de los 5 registros desde la posicion 5

print("Modified register values :" +str(client.read_input_registers(5, reg_nb=5)))
```

6.2.- Realice un programa cliente que escriba 5 registros sobre el servidor desde la dirección 10 y el resultado lo muestre el servidor con un mensaje indicativo de la acción realizada. El cliente también debe mostrar el resultado de la acción realizada, y para ello el cliente debe leer los 5 registros desde la dirección 10 en el servidor.

```
#!/usr/bin/env python3
from pyModbusTCP.server import ModbusServer, DataBank
from time import sleep
from random import uniform

server = ModbusServer("192.168.1.146", 23456, no_block=True) #Se crea un objeto
                                                              #servidor en la direccion ip
                                                              #de mi maquina en el
                                                              #puerto 23456

try:
 print("Start server...")#Notificacion para ver por consola el estado del servidor
 server.start()#Se inicia en el puerto 23456 el servicio servidor
 print("Server is on line")#Notificacion para ver por consola el estado del servidor

 state = [0,0,0,0,0] #Array con valores iniciales para algunos registros, se usará para
                     #comprobar si algun registro ha cambiado
 while True: #Bucle infinito para poner el servidor en escucha
   DataBank.set_words(0,[int(uniform(0,100))]) #Se generan numeros aleatorios entre 0 y
                                               #100 para rellenar los registros del servidor

   for i in range(10,15): #Bucle que itera sobre los numeros de registro que se pretenden
                          #modificar para poder comprobar si alguno ha cambiado
     if state[i-10] != DataBank.get_words(i) : #Comprobar si el registro ha cambiado con
                                               #respecto a su valor inicial
       state[i-10] = DataBank.get_words(i) #Si ha cambiado actualizamos el valor del array
                                           #con valores iniciales
       print("value register: " + str(i)  + " has changed to: " + str(state[i-10]))#Notificamos por
                                                                                  #la consola del
                                                                                  #servidor el
                                                                                  #cambio

       sleep(0.5)
except: #Gestion de errores y excepciones
 print("Shutdown server...")
 server.stop()
 print("Server is offline")
```

6.3.- Realice un programa cliente que escriba 5 registros sobre el servidor desde la dirección 15 y el resultado lo muestre el cliente y el servidor con un mensaje indicativo de la acción realizada tanto del lado del cliente como del servidor. El programa cliente debe realizar las siguientes verificaciones al leer los 5 registros desde el servidor.

6.3.1.- Indicar para cada registro si el número introducido es par o impar.

```
#!/usr/bin/env python3

from operator import index
from pyModbusTCP.client import ModbusClient

client = ModbusClient(host="192.168.1.146", port=23456)
client.open()
#Lee los valores iniciales de los 5 registros desde la posicion 15
print("Initial register values :" + str(client.read_input_registers(15, reg_nb=5)))
#Escribe 4 registors desde la direccion 15 hasta la  19 con valores 3,2,1,3,4 respectivamente
client.write_multiple_registers(15,[3,2,1,3,4])
#Itera sobre los valores de los 5 registros siguientes a la posicion 15 y comprueba si son impares
for indx,elem in enumerate(client.read_input_registers(15, reg_nb=5)):
    if(  elem % 2 ): print("Register number " + str(indx +15) + " is odd with a value of: " + str(elem))
    else: print("Register number " + str(indx +15) + " is even with a value of: " + str(elem))
```

6.3.2.- Indicar si la suma de los 5 registros introducidos es un número par o impar.

```
#!/usr/bin/env python3

from operator import index
from pyModbusTCP.client import ModbusClient

client = ModbusClient(host="192.168.1.146", port=23456)
suma=0 # Variable para contabilizar la suma de registros
client.open()

#Lee los valores iniciales de los 5 registros desde la posicion 15
print("Initial register values :" + str(client.read_input_registers(15, reg_nb=5)))

#Escribe 4 registors desde la direccion 15 hasta la 19 con valores 3,2,1,3,4 respectivamente
client.write_multiple_registers(15,[3,2,1,3,4])

#Itera sobre los valores de los 5 registros siguientes a la posicion 15 y comprueba si son impares
if(sum(client.read_input_registers(15, reg_nb=5))%2): print("La suma de los registros es impar y
vale: "+ str(sum(client.read_input_registers(15, reg_nb=5))))
else: print("La suma de los registros es par y vale: "+ str(sum(client.read_input_registers(15,
reg_nb=5))))
```