

PROTOCOLOS DE INTERNET

Trabajo Práctico N.º1 C :Modelo Cliente-Servidor

1 CUATRIMESTRE DE 2023

Trabajo Práctico N° 1

Grupo: 17

Profesor: Javier Ouret

Integrantes:

Nº	Apellido y Nombre	Carrera	Legajo	E mail
1	Cortina, Tomás	Ing. Informática	151803314	tomascortina00@uca.edu.ar
2	Denti, Mateo	Ing. Informática	152151542	mateo.denti@outlook.com.ar
3	Fantino, Thiago	Ing. Informática	152151054	thiagofantino@uca.edu.ar
4	Soriano, Máximo	Ing. Informática	152154633	maximo.soriano170103@gmail.com

PDI - TP 1 - C - Cliente Servidor utilizando Sockets en Python.

Para realizar programas Cliente Servidor con Python utilizamos 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 recursos 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
```

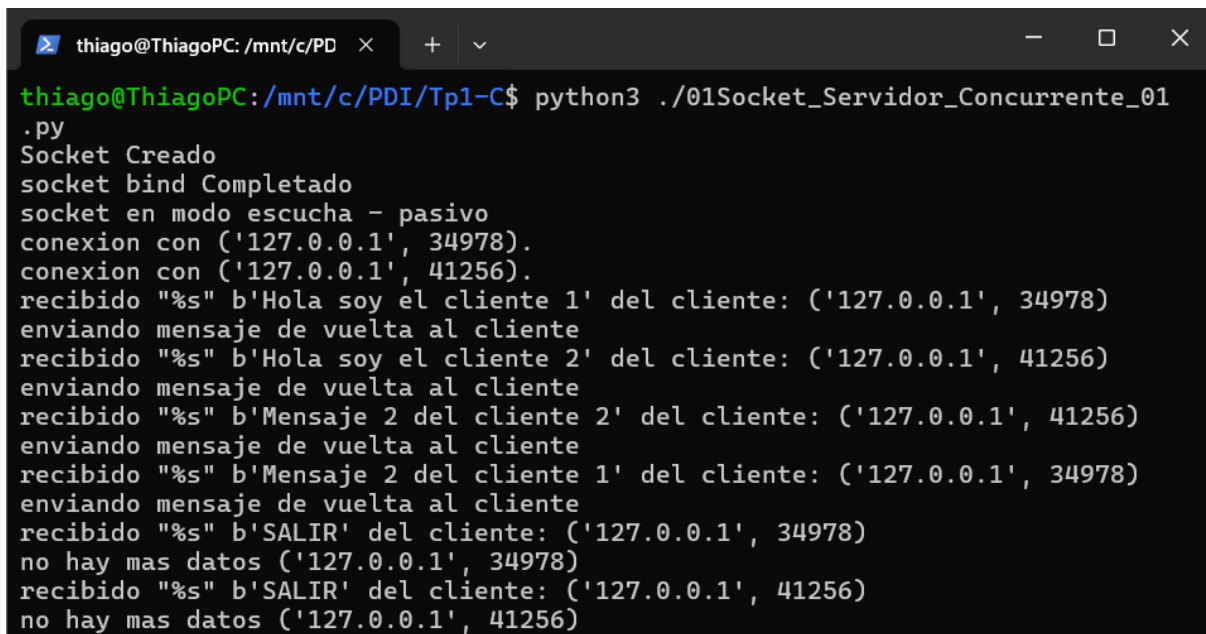
```

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()

```

Cliente-Servidor Concurrente

Servidor



```

thiago@ThiagoPC: /mnt/c/PD x + v
thiago@ThiagoPC:/mnt/c/PDI/Tp1-C$ python3 ./01Socket_Servidor_Concurrente_01
.py
Socket Creado
socket bind Completado
socket en modo escucha - pasivo
conexion con ('127.0.0.1', 34978).
conexion con ('127.0.0.1', 41256).
recibido "%s" b'Hola soy el cliente 1' del cliente: ('127.0.0.1', 34978)
enviando mensaje de vuelta al cliente
recibido "%s" b'Hola soy el cliente 2' del cliente: ('127.0.0.1', 41256)
enviando mensaje de vuelta al cliente
recibido "%s" b'Mensaje 2 del cliente 2' del cliente: ('127.0.0.1', 41256)
enviando mensaje de vuelta al cliente
recibido "%s" b'Mensaje 2 del cliente 1' del cliente: ('127.0.0.1', 34978)
enviando mensaje de vuelta al cliente
recibido "%s" b'SALIR' del cliente: ('127.0.0.1', 34978)
no hay mas datos ('127.0.0.1', 34978)
recibido "%s" b'SALIR' del cliente: ('127.0.0.1', 41256)
no hay mas datos ('127.0.0.1', 41256)

```

El server crea el socket, hace el bind y lo pone en modo escucha esperando los mensajes de los clientes.

Una vez recibe un mensaje del cliente, responde con el día y la hora en la que el mensaje fue recibido. También, responde con el tiempo que el cliente lleva conectado.

Cliente 1

```
thiago@ThiagoPC: /mnt/c/PD × thiago@ThiagoPC: /mnt/c/PDI × + v
thiago@ThiagoPC: /mnt/c/PDI/Tp1-C$ python3 ./01Socket_Cliente_02.py
conectando a %s puerto %s ('localhost', 6667)
Paso: 0
Servidor: Conectado con cliente
Ingrese mensaje ('SALIR' para terminar): Hola soy el cliente 1
enviando "%rb" b'Hola soy el cliente 1'
recibiendo "%s" b'09/05/2024 14:44:16 108 secs'
Paso: 1
Paso: 1
Servidor: Conectado con cliente
Ingrese mensaje ('SALIR' para terminar): Mensaje 2 del cliente 1
enviando "%rb" b'Mensaje 2 del cliente 1'
recibiendo "%s" b'09/05/2024 14:44:42 133 secs'
Paso: 2
Paso: 2
Servidor: Conectado con cliente
Ingrese mensaje ('SALIR' para terminar): SALIR
enviando "%rb" b'SALIR'
recibiendo "%s" b'146 secs'
Paso: 3
Tiempo que estuvo el cliente conectado al servidor: b'146 secs'
cerrando socket
thiago@ThiagoPC: /mnt/c/PDI/Tp1-C$
```

El cliente 1 se conecta al servidor y luego le envía un mensaje. El servidor le devuelve la fecha y el tiempo que lleva conectado el cliente.

Para terminar la conexión se envía SALIR y finalmente el servidor le responde con el tiempo que duró la conexión.

Cliente 2

```
thiago@ThiagoPC: /mnt/c/PDI x thiago@ThiagoPC: /mnt/c/PD x + v
thiago@ThiagoPC:/mnt/c/PDI/Tp1-C$ python3 01Socket_Cliente_02.py
conectando a %s puerto %s ('localhost', 6667)
Paso: 0
Servidor: Conectado con cliente
Ingrese mensaje ('SALIR' para terminar): Hola soy el cliente 2
enviando "%rb" b'Hola soy el cliente 2'
recibiendo "%s" b'09/05/2024 14:44:25 36 secs'
Paso: 1
Paso: 1
Servidor: Conectado con cliente
Ingrese mensaje ('SALIR' para terminar): Mensaje 2 del cliente 2
enviando "%rb" b'Mensaje 2 del cliente 2'
recibiendo "%s" b'09/05/2024 14:44:34 44 secs'
Paso: 2
Paso: 2
Servidor: Conectado con cliente
Ingrese mensaje ('SALIR' para terminar): SALIR
enviando "%rb" b'SALIR'
recibiendo "%s" b'70 secs'
Paso: 3
Tiempo que estuvo el cliente conectado al servidor: b'70 secs'
cerrando socket
thiago@ThiagoPC:/mnt/c/PDI/Tp1-C$
```

El cliente 2 se conecta al servidor y luego le envía un mensaje. El servidor le devuelve la fecha y el tiempo que lleva conectado el cliente.
Para terminar la conexión se envía SALIR y finalmente el servidor le responde con el tiempo que duró la conexión.

Cliente-Servidor con Concurrencia Aparente

Servidor

```
thiago@ThiagoPC: /mnt/c/PD × + v
thiago@ThiagoPC:/mnt/c/PDI/TP1-C$ python3 ./02Socket_Servidor_Select.py
iniciando en localhost port 10000
esperando el próximo evento
  conexión desde: ('127.0.0.1', 38180)
esperando el próximo evento
  conexión desde: ('127.0.0.1', 38194)
  recibido b'Este mensaje ' desde ('127.0.0.1', 38180)
esperando el próximo evento
  recibido b'Este mensaje ' desde ('127.0.0.1', 38194)
  enviando b'Este mensaje ' a ('127.0.0.1', 38180)
esperando el próximo evento
  ('127.0.0.1', 38180) cola vacía
  enviando b'Este mensaje ' a ('127.0.0.1', 38194)
esperando el próximo evento
  ('127.0.0.1', 38194) cola vacía
esperando el próximo evento
  recibido b'es enviado ' desde ('127.0.0.1', 38180)
esperando el próximo evento
  recibido b'es enviado ' desde ('127.0.0.1', 38194)
  enviando b'es enviado ' a ('127.0.0.1', 38180)
esperando el próximo evento
  ('127.0.0.1', 38180) cola vacía
  enviando b'es enviado ' a ('127.0.0.1', 38194)
esperando el próximo evento
  ('127.0.0.1', 38194) cola vacía
esperando el próximo evento
  recibido b'en partes.' desde ('127.0.0.1', 38180)
esperando el próximo evento
  recibido b'en partes.' desde ('127.0.0.1', 38194)
  enviando b'en partes.' a ('127.0.0.1', 38180)
esperando el próximo evento
  ('127.0.0.1', 38180) cola vacía
  enviando b'en partes.' a ('127.0.0.1', 38194)
esperando el próximo evento
  ('127.0.0.1', 38194) cola vacía
esperando el próximo evento
  cerrando... ('127.0.0.1', 38194)
  cerrando... ('127.0.0.1', 38194)
esperando el próximo evento
  conexión desde: ('127.0.0.1', 35630)
esperando el próximo evento
```

```
esperando el próximo evento
  conexión desde: ('127.0.0.1', 35636)
  recibido b'Este mensaje ' desde ('127.0.0.1', 35630)
esperando el próximo evento
  recibido b'Este mensaje ' desde ('127.0.0.1', 35636)
  enviando b'Este mensaje ' a ('127.0.0.1', 35630)
esperando el próximo evento
  ('127.0.0.1', 35630) cola vacía
  enviando b'Este mensaje ' a ('127.0.0.1', 35636)
esperando el próximo evento
  ('127.0.0.1', 35636) cola vacía
esperando el próximo evento
  recibido b'es enviado ' desde ('127.0.0.1', 35630)
esperando el próximo evento
  recibido b'es enviado ' desde ('127.0.0.1', 35636)
  enviando b'es enviado ' a ('127.0.0.1', 35630)
esperando el próximo evento
  ('127.0.0.1', 35630) cola vacía
  enviando b'es enviado ' a ('127.0.0.1', 35636)
esperando el próximo evento
  ('127.0.0.1', 35636) cola vacía
esperando el próximo evento
  recibido b'en partes.' desde ('127.0.0.1', 35630)
esperando el próximo evento
  recibido b'en partes.' desde ('127.0.0.1', 35636)
  enviando b'en partes.' a ('127.0.0.1', 35630)
esperando el próximo evento
  ('127.0.0.1', 35630) cola vacía
  enviando b'en partes.' a ('127.0.0.1', 35636)
esperando el próximo evento
  ('127.0.0.1', 35636) cola vacía
esperando el próximo evento
  cerrando... ('127.0.0.1', 35636)
  cerrando... ('127.0.0.1', 35636)
esperando el próximo evento
```

El server crea el socket, hace el bind y lo pone en modo escucha esperando los mensajes de los clientes.

Una vez recibe un mensaje del cliente, éste lo pone en una cola y al responder, lo elimina de la cola.

Cliente 1

```
thiago@ThiagoPC: /mnt/c/PDI x thiago@ThiagoPC: /mnt/c/PDI x + v
thiago@ThiagoPC:/mnt/c/PDI/Tp1-C$ python3 ./02Socket_Cliente_Select.py
conectando a localhost puerto 10000
('127.0.0.1', 38180): enviando b'Este mensaje '
('127.0.0.1', 38194): enviando b'Este mensaje '
('127.0.0.1', 38180): recibido b'Este mensaje '
('127.0.0.1', 38194): recibido b'Este mensaje '
('127.0.0.1', 38180): enviando b'es enviado '
('127.0.0.1', 38194): enviando b'es enviado '
('127.0.0.1', 38180): recibido b'es enviado '
('127.0.0.1', 38194): recibido b'es enviado '
('127.0.0.1', 38180): enviando b'en partes.'
('127.0.0.1', 38194): enviando b'en partes.'
('127.0.0.1', 38180): recibido b'en partes.'
('127.0.0.1', 38194): recibido b'en partes.'
thiago@ThiagoPC:/mnt/c/PDI/Tp1-C$ |
```

El cliente 1 se conecta al servidor y luego le envía un mensaje.

Cliente 2

```
thiago@ThiagoPC: /mnt/c/PDI x thiago@ThiagoPC: /mnt/c/PDI x + v
thiago@ThiagoPC:/mnt/c/PDI/Tp1-C$ python3 ./02Socket_Cliente_Select.py
conectando a localhost puerto 10000
('127.0.0.1', 35630): enviando b'Este mensaje '
('127.0.0.1', 35636): enviando b'Este mensaje '
('127.0.0.1', 35630): recibido b'Este mensaje '
('127.0.0.1', 35636): recibido b'Este mensaje '
('127.0.0.1', 35630): enviando b'es enviado '
('127.0.0.1', 35636): enviando b'es enviado '
('127.0.0.1', 35630): recibido b'es enviado '
('127.0.0.1', 35636): recibido b'es enviado '
('127.0.0.1', 35630): enviando b'en partes.'
('127.0.0.1', 35636): enviando b'en partes.'
('127.0.0.1', 35630): recibido b'en partes.'
('127.0.0.1', 35636): recibido b'en partes.'
thiago@ThiagoPC:/mnt/c/PDI/Tp1-C$
```

El cliente 2 se conecta al servidor y luego le envía un mensaje.