# PDI - TP 1 - C

Alumnos:
Alonso Facundo Nahuel
Alvarez Poli Bautista
Kloster Agustin

*Consignas:*
*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 de los programas (agregar opción de pregunta al usuario para cerrar los clientes).*

Solucion:

## Resultados

En el cliente

```
Conexión exitosa - Fecha:  2024-05-13 12:14:01.025265
Conectando a localhost puerto 10000
enviando b'Parte 1 del mensaje'
enviando b'Parte 2 del mensaje'
Respuesta: b'Parte 1 del mensajeParte 2 del mensaje'
Ingrese ENTER para finalizar la conexión
Conexión cerrada - Fecha:  2024-05-13 12:14:05.349042
Duración (hh:mm:ss):  0:00:04.323777
cerrando socket
```

En el servidor

```
iniciando en localhost port 10000
Esperando el próximo evento
Conexión desde ('127.0.0.1', 57382) (2024-05-13 12:14:01.025343)
Esperando el próximo evento
  recibido b'Parte 1 del mensaje' desde ('127.0.0.1', 57382)
Esperando el próximo evento
 enviando b'Parte 1 del mensaje' a ('127.0.0.1', 57382)
Esperando el próximo evento
   ('127.0.0.1', 57382) cola vacía
Esperando el próximo evento
  recibido b'Parte 2 del mensaje' desde ('127.0.0.1', 57382)
Esperando el próximo evento
 enviando b'Parte 2 del mensaje' a ('127.0.0.1', 57382)
Esperando el próximo evento
   ('127.0.0.1', 57382) cola vacía
Esperando el próximo evento
Finalizando conexión del cliente ('127.0.0.1', 57382) (2024-05-13 12:14:05.349114)
Duración: 0:00:04.323771
Esperando el próximo evento
```

## Codigo

Para **concurrencia aparente**, con select, el servidor va a ser

```python
import select
import socket
import sys
import queue
import datetime

# 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 {} port {}'.format(*dir_servidor),
      file=sys.stderr)
servidor.bind(dir_servidor)

# Escucho conexiones entrantes
servidor.listen(5)

# Sockect que espero leer
entradas = [servidor]

# Sockets que espero enviar
salidas = []
```

```
cola_mensajes = {}

while entradas:

    # Espero a que al menos uno de los sockets este listo para ser procesado

    print('Esperando el próximo evento', file=sys.stderr)
    readable, writable, exceptional = select.select(entradas,
                                                    salidas,
                                                    entradas)

    if not (readable or writable or exceptional):
        print('  tiempo excedido....',
              file=sys.stderr)
        continue
    # Manejo entradas
    for s in readable:

        if s is servidor:
            # Un socket "leíble" está listo para aceptar conexiones
            con, dir_cliente = s.accept()
            connStart = datetime.datetime.now()
            print('Conexión desde {} ({})'.format( *args: dir_cliente,connStart),
                  file=sys.stderr)
            con.setblocking(0)
            entradas.append(con)
```

```
            # Le asigno a la conexión una cola en la cuál quiero enviar
            cola_mensajes[con] = queue.Queue()

        else:
            data = s.recv(1024)
            if data:
                # Un socket leíble tiene datos
                print('  recibido {!r} desde {}'.format(
                    *args: data, s.getpeername()), file=sys.stderr,
                )
                cola_mensajes[s].put(data)
                # Agrego un canal de salida para la respuesta
                if s not in salidas:
                    salidas.append(s)
            else:
                # Si está vacío lo interpreto como una conexión a cerrar
                connClose = datetime.datetime.now()
                print('Finalizando conexión del cliente {} ({})\nDuración: {}'.format(
                    *args: dir_cliente,
                    connClose,
                    (connClose-connStart)),
                      file=sys.stderr)
                # dejo de escuchar en la conexión
                if s in salidas:
                    salidas.remove(s)
                entradas.remove(s)
                s.close()
```

```
115
116                 # Rremueve mensaje de la cola
117                 del cola_mensajes[s]
118         # Administro salidas
119         for s in writable:
120             try:
121                 next_msg = cola_mensajes[s].get_nowait()
122             except queue.Empty:
123                 # No hay mensaje en espera. Dejo de controlar para posibles escrituras
124
125                 print('  ', s.getpeername(), 'cola vacía',
126                       file=sys.stderr)
127                 salidas.remove(s)
128             else:
129                 print(' enviando {!r} a {}'.format( *args: next_msg,
130                         s.getpeername()),
131                         file=sys.stderr)
132                 s.send(next_msg)
133
134     # Administro condiciones excepcionales"
135         for s in exceptional:
136             print('excepción en', s.getpeername(),
137                   file=sys.stderr)
138             # Dejo de escuchar en las conexiones
139             entradas.remove(s)
140             if s in salidas:
141                 salidas.remove(s)
142             s.close()
```

```
144             # Remuevo cola de mensajes
145             del cola_mensajes[s]
146
```

El cliente va a ser

```python
import socket
import sys
import time
import datetime

# Crear un socket TCP/IP
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Conectar el socket al puerto en el cual el servidor está escuchando
dir_servidor = ('localhost', 10000)
print('Conectando a {} puerto {}'.format(*dir_servidor),file=sys.stderr)
sock.connect(dir_servidor)
connStart = datetime.datetime.now()
print('Conexión exitosa - Fecha: ',connStart)
time.sleep(0.5)
mensajes = [...]

try:
    # enviando datos
    for mensaje in mensajes:
        data = mensaje.encode()
        print('enviando {!r}'.format(data), file=sys.stderr)
        sock.sendall(data)
        time.sleep(1)

    # Respuesta
    data = sock.recv(1024)
    print('Respuesta: {!r}'.format(data), file=sys.stderr)

finally:
    input("Ingrese ENTER para finalizar la conexión")
    print('cerrando socket', file=sys.stderr)
    sock.close()
    connEnd = datetime.datetime.now()
    print('Conexión cerrada - Fecha: ',connEnd)
    print('Duración (hh:mm:ss): ',(connEnd-connStart))
```

El servidor con **concurrencia real**

```python
#servidor concurrente
import socket
import threading
import datetime
import time

host = "127.0.0.1"
port = 6667

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print ("Socket Creado")
sock.bind((host, port))
print ("Socket bind Completado")
sock.listen(1)
print ("Socket en modo escucha - pasivo")

1 usage
def proceso_hijo(*args): #*args valores de conexión y dirección de cliente devueltos por soc
    conn = args[0] #Conexión
    addr = args[1] #Dir Cliente
    connStart = datetime.datetime.now() # Almacena el timestamp de la apertura de conexión
    try:
        print('Conexion con {}.'.format(addr))
        conn.send("Conexión aceptada - Fecha: {}".format(connStart).encode('UTF-8'))

    while True:
        data = conn.recv(1024)
        print ('Recibido: ',data.decode("utf-8"))
        if data and data!=b'exit':
            print ('Enviando mensaje de vuelta al cliente')
            conn.sendall(data)
        else:
            print ('No hay mas datos', addr)
            break
inally:
    connEnd = datetime.datetime.now()
    connDuration = ("Fin de la conexión"
                    " - Duró: {} (hh:mm:ss)".format((connEnd-connStart))).encode('UTF-8')
    conn.send(connDuration)
    conn.close()
    print('Se cerró la conexión con: {}.'.format(addr))

 1:
onn, addr = sock.accept()
hreading.Thread(target=proceso_hijo, args=(conn, addr)).start()
```

Y el cliente con concurrencia real

```python
import socket
import sys
import time

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_dir = ('localhost', 6667)
print ('conectando a %s puerto %s',server_dir)
sock.connect(server_dir)

contador = 0
while contador <=3:
    # Enviando datos
    print("Paso:", contador)
    mensaje = b'contenidoaenviar'
    print('Enviando al servidor: {}'.format(mensaje.decode("utf-8")))
    sock.sendall(mensaje)
    data = sock.recv(1024)
    print('Respuesta recibida: {}'.format(data.decode("utf-8")))
    contador = contador + 1
    time.sleep(1)

input("Presione ENTER para finalizar la conexión")
print ('cerrando socket')
sock.recv(1024) # Se limpia el buffer de respuesta
sock.sendall(b'exit')
time.sleep(2)
print(sock.recv(1024).decode("utf-8"))
sock.close()
```

**Observación:** En ambos casos, para la aplicación del cliente, se utiliza un input() que aguarda hasta que el usuario ingrese cualquier cadena de texto - mediante la consola de comandos - para proseguir con el cierre de la conexión. Este cierre se hace, en primer lugar, enviando al servidor un mensaje con la palabra reservada “exit ”para indicarle el cierre de conexión, y este responde con la duración de la conexión. Finalmente se cierran los socket tanto del lado del servidor como del lado del cliente.