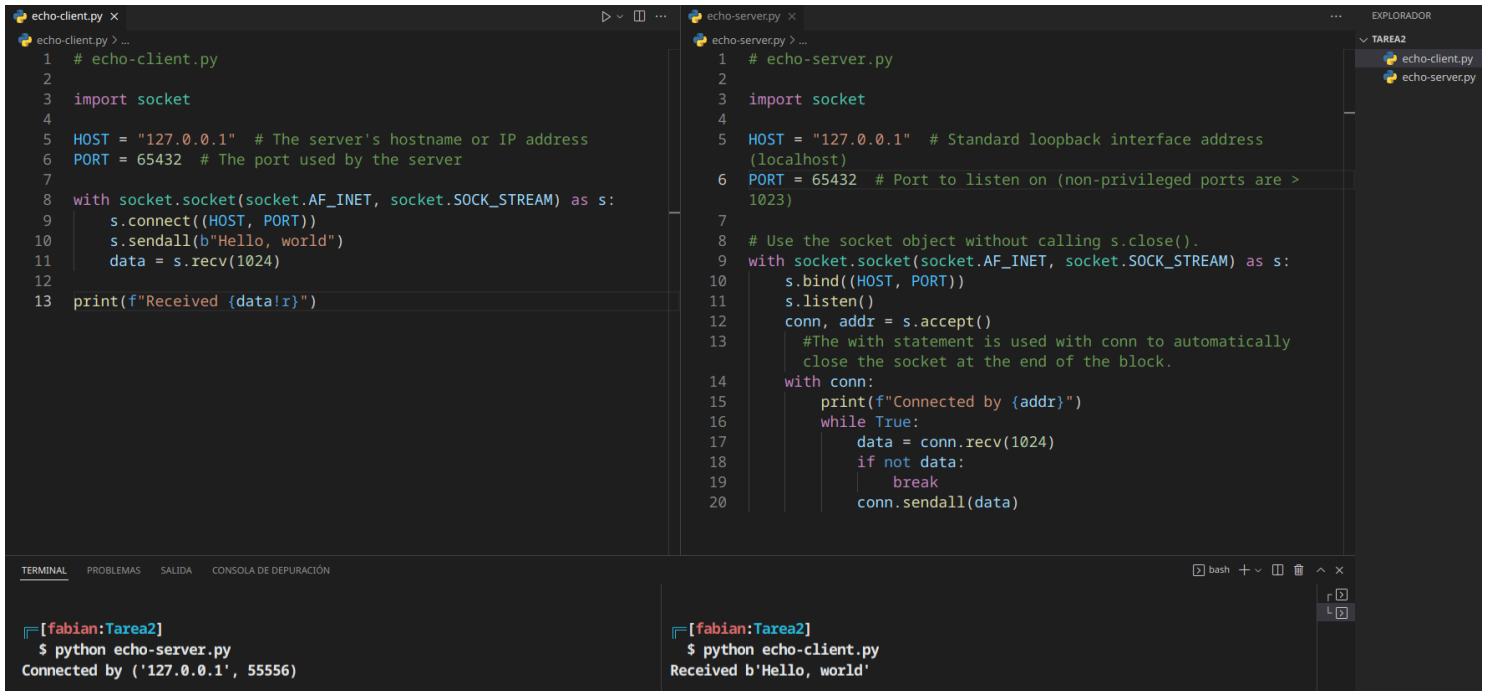


Universidad de Costa Rica  
Ciencias de la Computación e informática

Fabián Orozco Chaves - B95690  
Redes de Comunicación de Datos - Tarea 2

Ejemplos de sockets de los tutoriales

1) Python (Echo Server) - <https://realpython.com/python-sockets/>



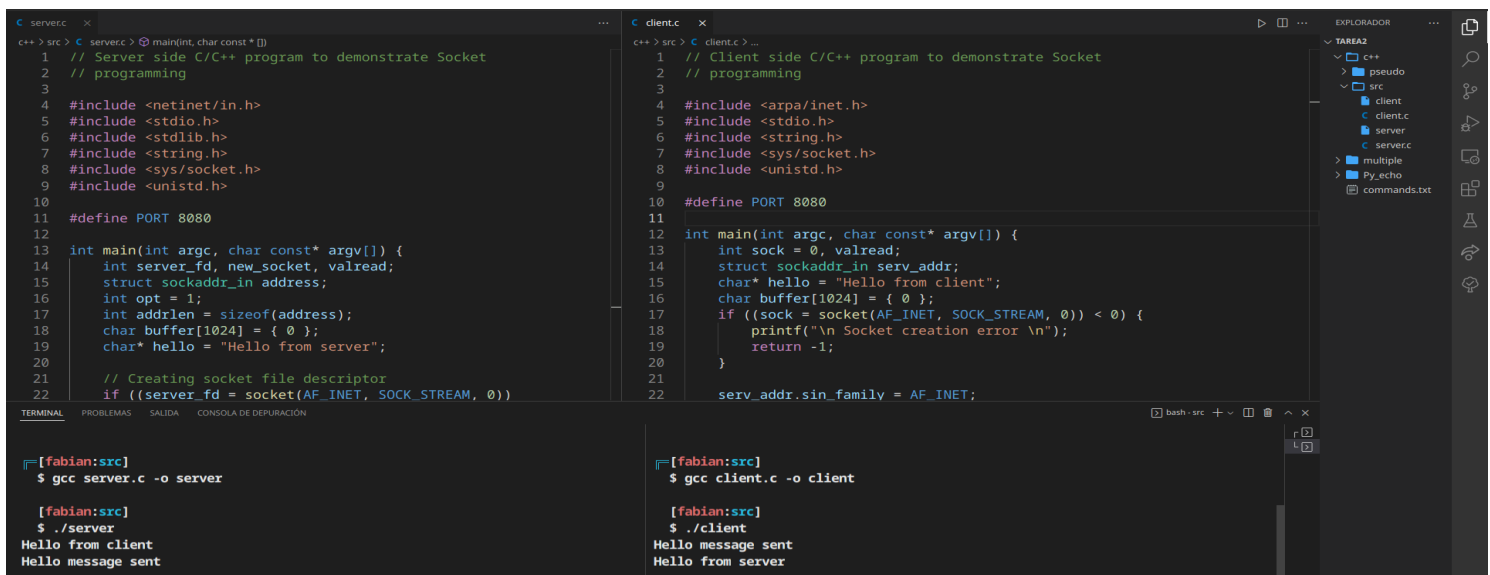
```
echo-client.py
1 # echo-client.py
2
3 import socket
4
5 HOST = "127.0.0.1" # The server's hostname or IP address
6 PORT = 65432 # The port used by the server
7
8 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
9     s.connect((HOST, PORT))
10    s.sendall(b"Hello, world")
11    data = s.recv(1024)
12
13 print(f"Received {data!r}")

echo-server.py
1 # echo-server.py
2
3 import socket
4
5 HOST = "127.0.0.1" # Standard loopback interface address (localhost)
6 PORT = 65432 # Port to listen on (non-privileged ports are > 1023)
7
8 # Use the socket object without calling s.close().
9 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
10    s.bind((HOST, PORT))
11    s.listen()
12    conn, addr = s.accept()
13    #The with statement is used with conn to automatically
14    #close the socket at the end of the block.
15    with conn:
16        print(f"Connected by {addr}")
17        while True:
18            data = conn.recv(1024)
19            if not data:
20                break
21            conn.sendall(data)
```

```
Terminal
[fabian:Tarea2]
$ python echo-server.py
Connected by ('127.0.0.1', 55556)

[fabian:Tarea2]
$ python echo-client.py
Received b'Hello, world'
```

2) C++ - <https://www.geeksforgeeks.org/socket-programming-cc/>



```
server.c
1 // Server side C/C++ program to demonstrate Socket
2 // programming
3
4 #include <netinet/in.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8 #include <sys/socket.h>
9 #include <unistd.h>
10
11 #define PORT 8080
12
13 int main(int argc, char const* argv[]) {
14     int server_fd, new_socket, valread;
15     struct sockaddr_in address;
16     int opt = 1;
17     int addrlen = sizeof(address);
18     char buffer[1024] = { 0 };
19     char* hello = "Hello from server";
20
21     // Creating socket file descriptor
22     if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
23         printf("\n Socket creation error \n");
24         return -1;
25     }
26
27     serv_addr.sin_family = AF_INET;
28     serv_addr.sin_port = htons(PORT);
29
30     // Creating socket file descriptor
31     if ((new_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
32         printf("\n Socket creation error \n");
33         return -1;
34     }
35
36     // Connecting to the server
37     if (connect(new_socket, &serv_addr, sizeof(serv_addr)) < 0) {
38         printf("\n Error connecting to the server \n");
39         return -1;
40     }
41
42     printf("Socket successfully created\n");
43
44     // Sending data to the server
45     send(new_socket, hello, strlen(hello), 0);
46
47     printf("Message sent\n");
48
49     // Receiving data from the server
50     valread = read(server_fd, buffer, 1024);
51     if (valread > 0) {
52         printf("Message received: %s\n", buffer);
53     }
54
55     return 0;
56 }
```

```
client.c
1 // Client side C/C++ program to demonstrate Socket
2 // programming
3
4 #include <arpa/inet.h>
5 #include <stdio.h>
6 #include <string.h>
7 #include <sys/socket.h>
8 #include <unistd.h>
9
10 #define PORT 8080
11
12 int main(int argc, char const* argv[]) {
13     int sock = 0, valread;
14     struct sockaddr_in serv_addr;
15     char* hello = "Hello from client";
16     char buffer[1024] = { 0 };
17     if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
18         printf("\n Socket creation error \n");
19         return -1;
20     }
21
22     serv_addr.sin_family = AF_INET;
23     serv_addr.sin_port = htons(PORT);
24
25     // Connecting to the server
26     if (connect(sock, &serv_addr, sizeof(serv_addr)) < 0) {
27         printf("\n Error connecting to the server \n");
28         return -1;
29     }
30
31     printf("Socket successfully created\n");
32
33     // Sending data to the server
34     send(sock, hello, strlen(hello), 0);
35
36     printf("Message sent\n");
37
38     // Receiving data from the server
39     valread = read(server_fd, buffer, 1024);
40     if (valread > 0) {
41         printf("Message received: %s\n", buffer);
42     }
43
44     return 0;
45 }
```

```
Terminal
[fabian:src]
$ gcc server.c -o server

[fabian:src]
$ ./server
Hello from client
Hello message sent

[fabian:src]
$ gcc client.c -o client

[fabian:src]
$ ./client
Hello message sent
Hello from server
```

### 3) Python & C++ - <https://tinyurl.com/socketPyAndC>

The image shows a Visual Studio Code editor with two C++ files: `Receiver.cpp` and `Transmitter.cpp`. Both files use `WSASession` for socket communication. The receiver listens on port 8888, and the transmitter sends "hello world" to 127.0.0.1 on port 8888. The terminal shows the compilation and execution of both programs, with the receiver displaying the received data.

```
C++ Receiver.cpp > ...
1 // C++ UDP Receiver
2
3 #include "Network.h"
4 #include <iostream>
5
6 #pragma once
7
8 int main() {
9     int PORT = 8888;
10
11     try {
12         WSASession Session;
13         UDPSocket Socket;
14         char buffer[100];
15
16         Socket.Bind(PORT);
17         while (true) {
18             sockaddr_in add = Socket.RecvFrom(buffer, sizeof(buffer));
19             std::string innut(buffer);
20         }
21     }
22 }
```

```
C++ Transmitter.cpp > ...
1 // C++ UDP Transmitter
2
3 #include "Network.h"
4 #include <iostream>
5
6 #pragma once
7
8 int main() {
9     std::string IP = "127.0.0.1";
10    int PORT = 8888;
11
12    try {
13        WSASession Session;
14        UDPSocket Socket;
15        std::string data = "hello world";
16        char buffer[100];
17
18        while (true) {
19            std::cout << "Enter data to transmit : " << std::endl;
20            std::string data;
21            if (data == "exit") break;
22            Socket.Sendto(buffer, data, IP, PORT);
23        }
24    }
25 }
```

Terminal Output (Left):

```
[Estudiante:Both]
$ g++ Receiver.cpp -lws2_32 -o receiver
Receiver.cpp:6:9: warning: #pragma once in main file
6 | #pragma once
  | ~~~~~
[Estudiante:Both]
$ ./receiver.exe
Data received: Hello World
Data received: Fabian Orozco end program
Data received:
[Estudiante:Both]
$
```

Terminal Output (Right):

```
[Estudiante:Both]
$ g++ Transmitter.cpp -lws2_32 -o transmitter
Transmitter.cpp:6:9: warning: #pragma once in main file
6 | #pragma once
  | ~~~~~
[Estudiante:Both]
$ ./transmitter.exe
Enter data to transmit :
Hello World
Enter data to transmit :
Fabian Orozco end program
Enter data to transmit :
[Estudiante:Both]
$
```

Utilizando el **receiver** de Python.

The image shows a Visual Studio Code editor with a Python file `PY_Receiver.py` and a C++ file `Transmitter.cpp`. The Python receiver listens on port 8888 and prints the received data. The C++ transmitter sends "hello world" to 127.0.0.1 on port 8888. The terminal shows the execution of both programs, with the Python receiver displaying the received data.

```
PY_Receiver.py > ...
29 else:
30     return data
31
32 def main():
33     print ("[Fabian] Inicia Py_Receiver\n")
34
35     my_sock = initUDP(IP, PORT)
36     msg_received = "init_true"
37
38     while (msg_received):
39
40         msg_received = readUDP( my_sock )
41         print (msg_received.decode("utf-8"))
42
43     print ("\n[Fabian] Termina Py_Receiver\n")
44     main()
45
```

```
C++ Transmitter.cpp > ...
1 // C++ UDP Transmitter
2
3 #include "Network.h"
4 #include <iostream>
5
6 #pragma once
7
8 int main() {
9     std::string IP = "127.0.0.1";
10    int PORT = 8888;
11
12    try {
13        WSASession Session;
14        UDPSocket Socket;
15        std::string data = "hello world";
16        char buffer[100];
17
18        while (true) {
19            std::cout << "Enter data to transmit : " << std::endl;
20            std::string data;
21            if (data == "exit") break;
22            Socket.Sendto(buffer, data, IP, PORT);
23        }
24    }
25 }
```

Terminal Output (Left):

```
[Estudiante:Both]
$ python PY_Receiver.py
[Fabian] Inicia Py_Receiver

Hello
World

[Fabian] Termina Py_Receiver
$
```

Terminal Output (Right):

```
[Estudiante:Both]
$ ./transmitter.exe
Enter data to transmit :
Hello
Enter data to transmit :
World
Enter data to transmit :
[Estudiante:Both]
$
```