
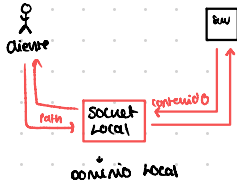


Sockets

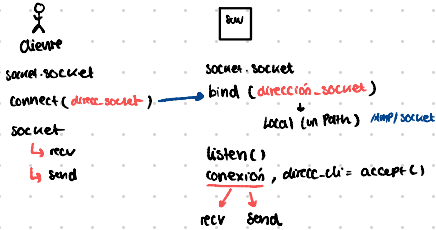


Sockets : Local

objetivo



Implementación



cliente

```
import os, sys, socket

def main():
    if len(sys.argv) < 2:
        print("Numero invalido de parametros")
        return 1

    path_fichero = sys.argv[1]

    with socket.socket(socket.AF_UNIX, socket.SOCK_STREAM, 0) as s:
        s.connect("/tmp/socket")

        s.send(path_fichero.encode())

        datos_bin = b''

        while True:
            datos = s.recv(1024)
            if not datos:
                break

            datos_bin += datos

        try:
            sys.stdout.write(datos_bin.decode('utf-8'))
        except UnicodeDecodeError:
            sys.stdout = sys.stdout.detach()
            sys.stdout.write(datos_bin)

if __name__ == "__main__":
    main()
```

servidor

```
import os, time, socket

def leerfichero(path_fichero):
    with open(path_fichero, 'rb') as file:
        return file.read()

def main():
    try:
        os.unlink("/tmp/socket")
    except OSError:
        pass

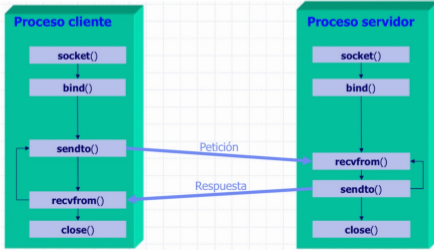
    with socket.socket(socket.AF_UNIX, socket.SOCK_STREAM, 0) as s:
        s.bind("/tmp/socket")

        while True:
            s.listen()

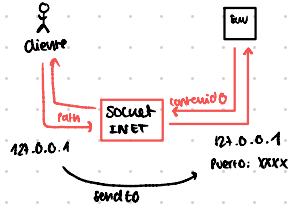
            conn, addr = s.accept()
            with conn:
                datos = conn.recv(1024).decode()
                conn.send(leerfichero(datos))

if __name__ == '__main__':
    main()
```

Socket : DGRAM (UDP)



objetivo



cliente

```
import socket
import sys

def main():
    direcc_serv = '127.0.0.1'
    puerto = input("Introduzca el puerto de su servidor: ")
    cli_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    mensaje = input("Introduce la dirección del archivo a enviar: ")
    cli_socket.sendto(mensaje.encode(), (direcc_serv, int(puerto)))

    direcc_local = cli_socket.getsockname()
    print(f'la dirección local es {direcc_local}')

    respuesta = cli_socket.recv(1024)
    print("Respuesta del servidor:\n", respuesta.decode())

if __name__ == "__main__":
    main()
```

Socket
sendto
recvfrom

servidor

```
import socket
import os
import datetime
import sys

def main():
    # direcc_socket = '0.0.0.0' # Escucha a todas las direcciones
    direcc_socket = '127.0.0.1'
    pr4_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    pr4_socket.bind((direcc_socket, 0))

    print("=====/n")
    direcc_local = pr4_socket.getsockname()
    print(f'la dirección local es {direcc_local}')

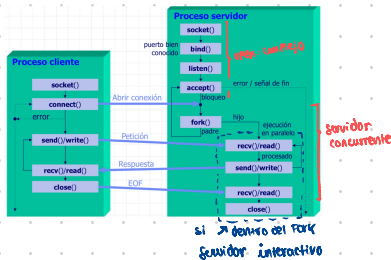
    while True:
        data, direcc_cli = pr4_socket.recvfrom(1024)

        with open(data.decode(), 'r') as file:
            contenido = file.read()
            pr4_socket.sendto(contenido.encode(), direcc_cli)

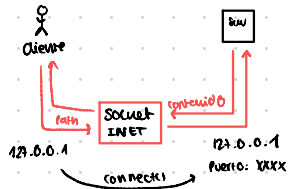
if __name__ == "__main__":
    main()
```

Socket
bind
recvfrom
sendto

SOCKETS : STREAM (TCP)



objetivo



cliente

```
import os, sys, socket

def main():
    if len(sys.argv) < 2:
        print("Numero invalido de parametros")
        return 1

    path_fichero = sys.argv[1]

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0) as s:
        s.connect(("0.0.0.0", 8086))

        s.send(path_fichero.encode())

        datos_bin = b''

        while True:
            datos = s.recv(1024)
            if not datos:
                break

            datos_bin += datos

        try:
            sys.stdout.write(datos_bin.decode('utf-8'))
        except UnicodeDecodeError:
            sys.stdout = sys.stdout.detach()
            sys.stdout.write(datos_bin)

if __name__ == "__main__":
    main()
```

servidor

```
import os, time, socket

def leerfichero(path_fichero):
    with open(path_fichero, 'rb') as file:
        return file.read()

def main(): # Delegar en un hijo para /dev/null
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0) as s:
        s.bind(("0.0.0.0", 8086))

        while True:
            s.listen()

            conn, addr = s.accept()

            with conn:
                datos = conn.recv(1024).decode()
                conn.send(leerfichero(datos))

if __name__ == '__main__':
    main()
```

⚠ No es concurrente ↗

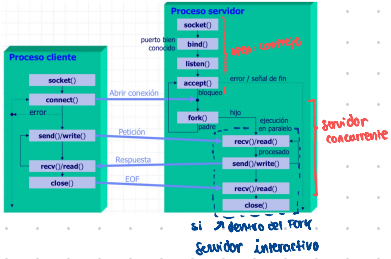
concurrente ↗ una vez aceptada conexión → delegamos fork

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind(("0.0.0.0", 8086))
    s.listen(5) # Permitir hasta 5 conexiones en espera

    while True:
        conn, addr = s.accept()
        print(f"Conexión entrante de {addr}")

        pid = os.fork()
        if pid == 0: # Proceso hijo
            s.close() # Cerrar el socket del padre en el hijo
            handle_connection(conn)
            os._exit(0)
        else: # Proceso padre
            conn.close() # Cerrar el socket del hijo en el padre
```

SOCKETs : STREAM (TCP)



concurrente :

delegamos a fork() ----> recv & sendall

servidor concurrente

```
import os
import socket

def leerfichero(path_fichero):
    with open(path_fichero, 'rb') as file:
        return file.read()

def handle_connection(conn):
    try:
        while True:
            datos = conn.recv(1024).decode()
            if not datos:
                break
            conn.sendall(leerfichero(datos))
    finally:
        conn.close()

def main():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind(("0.0.0.0", 8086))
        s.listen(5) # Permitir hasta 5 conexiones en espera

    while True:
        conn, addr = s.accept()
        print(f"Conexión entrante de {addr}")

        pid = os.fork()
        if pid == 0: # Proceso hijo
            s.close() # Cerrar el socket del padre en el hijo
            handle_connection(conn)
            os._exit(0)
        else: # Proceso padre
            conn.close() # Cerrar el socket del hijo en el padre

if __name__ == '__main__':
    main()
```

fork()

cliente

```
import os, sys, socket

def main():
    if len(sys.argv) < 2:
        print("Numero invalido de parametros")
        return 1

    path_fichero = sys.argv[1]

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0) as s:
        s.connect(("0.0.0.0", 8086))

        s.send(path_fichero.encode())

    datos_bin = b''

    while True:
        datos = s.recv(1024)
        if not datos:
            break

        datos_bin += datos

    try:
        sys.stdout.write(datos_bin.decode('utf-8'))
    except UnicodeDecodeError:
        sys.stdout = sys.stdout.detach()
        sys.stdout.write(datos_bin)

if __name__ == '__main__':
    main()
```

SOCKETS : STREAM (TCP)

uso del select