Stream sockets and multiplexing

Exercise 1 - Simple client

Implement a client that receives on the command line the IP address of a server and a text message, sends the message to the server, and prints the server's response on the standard output. Use `SOCK_STREAM` sockets in the `AF_INET` domain on port `5015`.

See file

Exercise 2 - Multiprocess server

Implement a server that, upon receiving a text message, replaces every small letter in it with the corresponding upper-case letter, and sends the transformed message back to the client. Use `SOCK_STREAM` sockets in the `AF_INET` domain on port `5015`. The structure of the server is based on multitasking. The initial process creates the listening socket and supervises the incoming connection requests. When a new connection is accepted, a new "communication" process is created to handle the connection until it is terminated. When the communication process terminates, the initial process receives the `SIGCHLD` signal.

The outline of the server is as follows:

```
Create a listening socket (system call socket())
Attach the socket to a local address (system call bind())
Prepare to receive connection requests (system call listen())
Install a handler for the SIGCHLD signal (system call signal())

while true do
    Wait for a new connection request (system call accept())
    Create a new communication process (c)
    (c) while not done do
    (c) Receive a request (system call read())
    (c) Handle the request or exit the loop if done
    (c) Send the response (system call write())
    (c) Close the connection (system call shutdown())
    (c) Close the connected socket (system call close())
```

Why is it necessary to handle the SIGCHLD signal in this application?

Exercise 3 - Single-process server

Implement the server from the previous exercise using a single process and multiplexing.

The outline of the server is as follows:

```
Create a listening socket (system call socket())
Attach the socket to a local address (system call bind())
```

```
Prepare to receive connection requests (system call listen())
Initialize the descriptor set and put the listening socket in it

while true do
    Wait for incoming data/events (system call select())
    if the listening socket is ready (function FD_ISSET())
        Accept the connection request (system call accept())
        Add the connected socket to the descriptor set (function FD_SET())
    for each ready connected socket (function FD_ISSET()) do
        Receive a request (system call read())
        if client disconnected
            Close the connection (system call shutdown())
            Close the connected socket (system call close())
            Remove the connected socket from the set (function FD_CLR())
        else
            Handle the request
            Send the response (system call write())
```