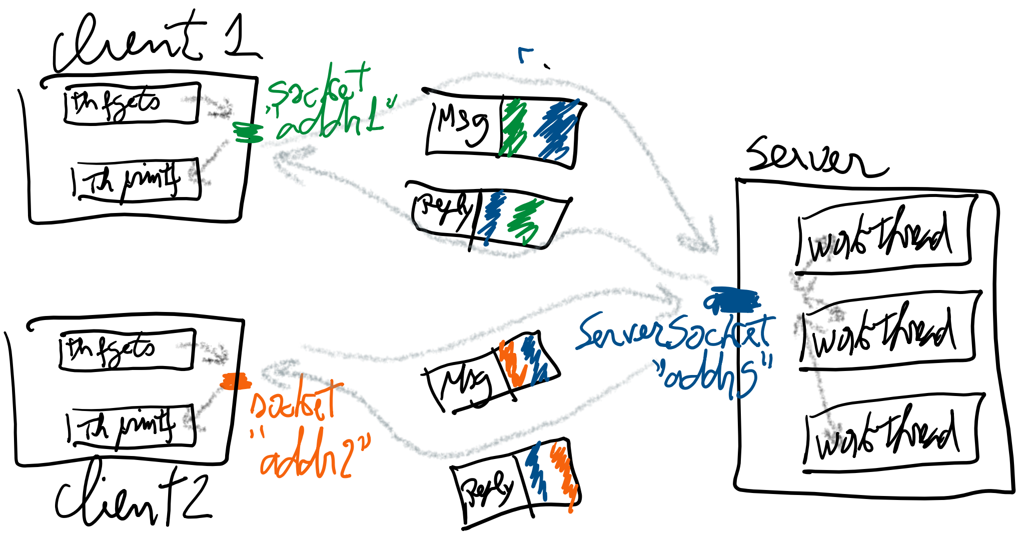
System Programming
7th Laboratory (31st March … 3rd April 2020)

In this laboratory students will implement a system that verifies if numbers generated by various clients are prime, but using two different types of sockets: datagram and stream. The system is composed of a multithreaded server and multiple clients as presented in the figures.
Each client will read from the keyboard integers that will be sent to the server. The server verifies if those numbers are prime and send back such information.

I

In the first exercise the clients will communicate with the server using datagram sockets, as illustrated in the figure:



Each client will have one sockets that will be used to send the numbers to the server and receive the reply:

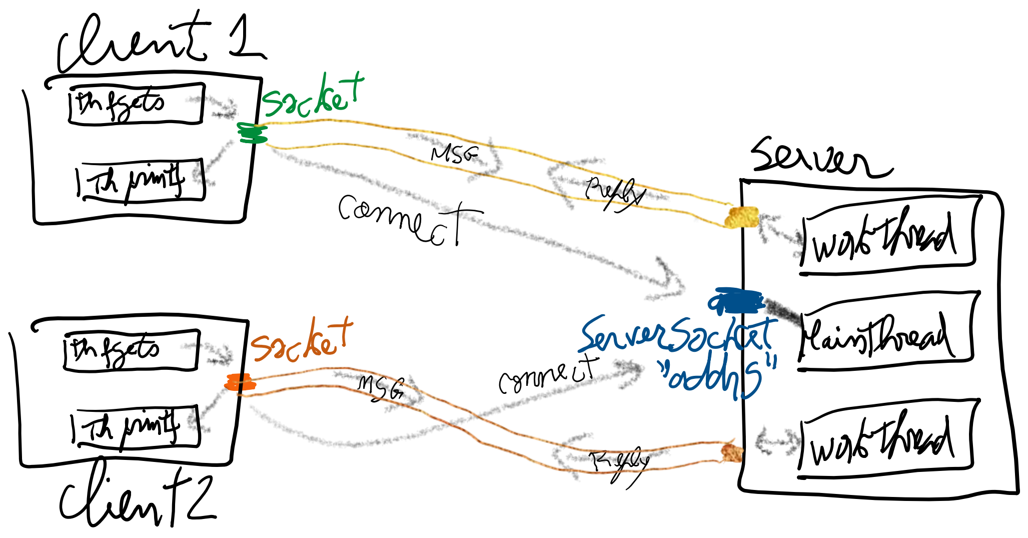
- The socket is of type datagram and should have an address assigned.
- The client will have one thread (**Th_fgets**) that will read numbers from the keyboard and send them to the server.
- The client will have another thread (**Th_printf**) that will read the server reply from the socked and print it on the screen.

The server will have one **server socket** and multiple threads:

- The **server socket** is of type datagram and should have an address
- The address of the **server socket** should be known by all clients (for instance defined in a .h file
- The messages received in the **server socket** are read and processed by the **workThreads**
- Every time the server receives a message from a new client a new **workThreads** is created.
- In the beginning there should be one **workThreads** and when the first client sends a message a new **workThreads** is created
- The various **workThreads** will all read (using the **recvfrom**) from the same thread.

I

Implement the previous system using stream sockets, following the next architecture:



Each client will have one stream socket that will be used communicate with the server. After the connection the client will use it to send the numbers to the server and receive the reply:

- The socket is of type stream
- Before creating the threads the client should coonect to the **server socket** using its local socket (**socket** or **socket**).
- The client will create one thread (**Th_fgets**) that will read numbers from the keyboard and send them to the server trough its local socket (**socket** or **socket**).
- The client will create another thread (**Th_printf**) that will read the server reply from its local socket (**socket** or **socket**) and print it on the screen.

The server will have one stream **server socket** and multiple threads:

- The **server socket** is of type stream and should have an address
- The address of the **server socket** should be known by all clients (for instance defined in a .h file
- At startup the server configure the **server socket** to receive connections.
- The **mainThread** will wait for connections of such **server socket.**
- When a client connects to the **server socket**, a new socket is automatically created and a new **workThreads** should be launched
- Each **workThreads** will be dedicated to each client and will read and write from the newly created socket.
- When the client disconnects the socket should be closed and the thread should be destroyed.

Things to define before starting to program

- address of the **server socket**

This location should be absolute (starting at the root of the file system).
A good place can be the ***/tmp/*** directory.
In Windows Subsystem for Linux (WSL)
it is fundamental to use the ***/tmp/*** directory!

- Address of the client socked (in Exercise **I**)

This location should be absolute (starting at the root of the file system).
A good place can be the ***/tmp/*** directory. In Windows Subsystem for Linux (WSL)
it is fundamental to use the ***/tmp/*** directory!

- Format of the data transferred in the various sockets (requests and replies)

SYSTEM CALLS

- socket
- bind
- sendto / recvfrom
- listen
- accept
- connect
- send / recv

REFERENCES

- http://tldp.org/LDP/lpg/node7.html (section 6.3)
- http://beej.us/guide/bgipc/html/multi/fifos.html