Exercise Sheet 3

Deadline: May 28, 2023 – 04:00 am CEST

Exercise 1 (Project „Online Room Manager“)

The previously developed IT system for room management at the university has already improved the situation a lot. However, in order to make the system accessible for all employees, it would be great to have a version with network access. Hence, the existing solution shall be extended by network functionality.

The IT department of the university would like to preserve the user interface from the existing solution as much as possible. Fortunately, the developers of the open source library „Room Management Library“ have specified an RPC protocol for the library. However, the only provide a client implementation (written in Python) but the server code is not yet ready.

In your git repository you will find a new directory `03`. Once again this directory contains the *Room Management Library* as a submodule. (*Note: Do not forget to run `git submodule init` and `git submodule update` first!*) The library directory (`libsrc`) also includes the header files containing the protocol specification. You will find the (updated) online documentation at `https://teaching.dahahm.de/roommannet/`. (*Note: The library API itself has been updated as well!*)

The subdirectory `src` contains a template which can be used as a starting point to implement the RPC server. Extend this application to implement a C program called `roommanagerserver` which expects the command line argument `-p PORT` to configure the listening port. (Again, the directory already contains a `Makefile` which builds the library and the server application.)

Per default the server should only listen on `localhost`, i.e., the IPv4 address `127.0.0.1`. That is a sane default because it prevents that services are accessible from the outside. Another command line argument, `-h IPADDRESS` shall be provided to listen at a different IP address.

Your task is to implement a server which makes the room manager library functions accessible via the roomman RPC protocol. I.e., a client sends a request to call a function which your server receives and decodes, is then calling the corresponding room management function, and sends the response back to the client.

The server may be implemented as a simple sequential server. The client might establish a new connection per RPC calls and your server should be capable to handle multiple requests in a row.

The subdirectory `client` contains a Python library to implement a client application and a ready-to-run client. You can call `python3 client/roomman_client.py` to connect the client to your server.

Hints

Familiarize yourself with the protocol to be implemented. Try to understand the required process flow and first make a plan what needs to be done to implement the RPCs. You can use pseudo code to draft the program flow.

Go step by step and test your solution after each step. Start by implementing the basic TCP server functionality and check whether the client can connect. Next, implement reception and decoding of the header, etc.

It might be advisable to split up functions into multiple C modules. If necessary you may need to add further `.c` and `.h` files in `src`. Do not forget these files to the git repository as well.

Hinweise:

- Erroneous or missing user input should be intercepted and return an error message. Your program should print a short usage description („RTFM text“).
- Wrong parameters (e.g., too long names for a building or a room) are rejected by the library. In these cases the server should create corresponding error messages and send them to the client.
- The encoding of integer types follow the network byte order (big endian).