Assignment 1

Socket Programming

Deadline: Monday, 26th October 2015, 9:00 am

Task 1.1: Organizational (0 Points)

This year we are managing all assignments and later projects using GitLab. GitLab is an open source git hosting and management system like github.com. Make yourself familiar with the GitLab environment running on https://laboratory.comsys.rwth-aachen.de. Register a new user. Once you successfully logged in. Create a group together with your lab partner, also make sure to add all advisors as a "Reporter".

Manage your source code using the GitLab supplied environment for all assignments. Simply create a new project for each assignment. To hand in your solutions for the assignments upload the git commit hash (see git log) of your final version into the L2P learning room together with a reference to your project.

Task 1.2: TCP Client (6 Points + *2 Points)

There is a server running on `laboratory.comsys.rwth-aachen.de:2345` using TCP sockets that is capable of telling "bad jokes". Your goal: Create a C Program that connects to this server using TCP and request a joke including your name. Follow this guideline:

1. Fetch server name and port as command line arguments.
2. Fetch a first name and last name from the user that he/she enters via the keyboard.
3. Connect to the server.
4. The server requires you to send a packet including the following:

Type	Length	Description
Unsigned Byte (uint8_t)	1 byte	Type == 1
Unsigned Byte (uint8_t)	1 byte	Length of first name
Unsigned Byte (uint8_t)	1 byte	Length of last name
String (char[], ASCII)	variable	String first name
String (char[], ASCII)	variable	String last name

   So a possible packet would be: $|0x01|0x05|0x06|C|h|u|c|k|N|o|r|r|i|s|$ (Note: the | is only for illustration to denotes a single byte). Use c-structs to create headers for your packets.

5. Upon correct reception of the data the server will try to answer with the following structure containing a Joke:

Type	Length	Description
Unsigned Byte (uint8_t)	1 byte	Type == 2
Unsigned Int (uint32_t, Big Endian)	4 byte	Length of the following Joke
String (char[], ASCII)	variable	String Joke

   Sometimes the server can't think of a funny joke, make sure to wait no longer than 3 seconds for an answer.

Teaching assistants:
Jan Rüth: `rueth@comsys.rwth-aachen.de`
Jó Ágila Bitsch: `jo.bitsch@cs.rwth-aachen.de`
Martin Serror: `martin.serror@comsys.rwth-aachen.de`

6. Close the socket and display the Joke

Note: Your program should compile with the following command:
`gcc -O2 -Wall -Wextra -Wshadow -pedantic -Werror -std=gnu99 joker_client.c -o joker_client`

-O2: Enable optimization level 2
-Wall: enable all warnings
-Wextra: enable some extra warnings not enabled by –Wall
-Wshadow: throw a warning when shadowing a variable, parameter or type
-pedantic: Issue all warnings demanded by the ISO C standard
-Werror: threat all warnings as errors (so your code must be warning free to compile)
-std=gnu99: Use GNU C99 standard
joker_client.c: Your c code
-o joker_client: name the executable joker_client

Your code should be well commented and structured. You should be able to explain its functionality in detail.

Hints:

1. Use c-structs for the packet headers and defines for the types like this:

```
#define JOKER_REQUEST_TYPE 1
#define JOKER_RESPONSE_TYPE 2

typedef struct {
    uint8_t type;
} __attribute__ ((__packed__)) joker_header;

typedef struct {
    uint8_t type;
    uint8_t len_first_name;
    uint8_t len_last_name;
} __attribute__ ((__packed__)) joker_request;

//...
```

2. Use clean types like `uint8_t` or `uint32_t` to ensure correct size. Find out why using `__attribute__ ((__packed__))` is usually a good idea for structs that are to be sent via network or could it be neglected here?

3. You are using streaming TCP sockets therefore packets will arrive in order. However, it is not guaranteed that all data is received using a single `recv` call, make sure to receive all data!

4. Take care of the endianness!

5. Although the server responds with a joke, sometimes it is so overexcited it will send you even more. Make sure to only display the joke.

6. Use the socket option `SO_RCVTIMEO` and the function `setsockopt` to realize the 3 seconds timeout. **Or better**: Don't use blocking sockets at all!

***Bonus:** The server is capable of using TCP Fast Open. Make your client ready for TFO and prove with a packet dump (in pcap format) that you indeed grabbed a TFO cookie and were able to send subsequent requests using this cookie to the server.

Teaching assistants:
Jan Rüth: `rueth@comsys.rwth-aachen.de`
Jó Ágila Bitsch: `jo.bitsch@cs.rwth-aachen.de`
Martin Serror: `martin.serror@comsys.rwth-aachen.de`

Task 1.3: TCP Server (4 Points + *1 Point)

Recreate the server functionality. Your jokes can be static or chosen from a set or you get them from somewhere else. However, your server should be able to handle at least 10 requests at the same time, use threads to accomplish this feature. You can write the server in a programming language of your choice though see * for additional info. Regardless of the programming language: structure your code well and don't forget to document it!

***Write your server in C to get 1 extra point.**

Make sure to upload the commit hash to the L^{2}P room before the deadline.

Teaching assistants:
Jan Rüth: rueth@comsys.rwth-aachen.de
Jó Ágila Bitsch: jo.bitsch@cs.rwth-aachen.de
Martin Serror: martin.serror@comsys.rwth-aachen.de