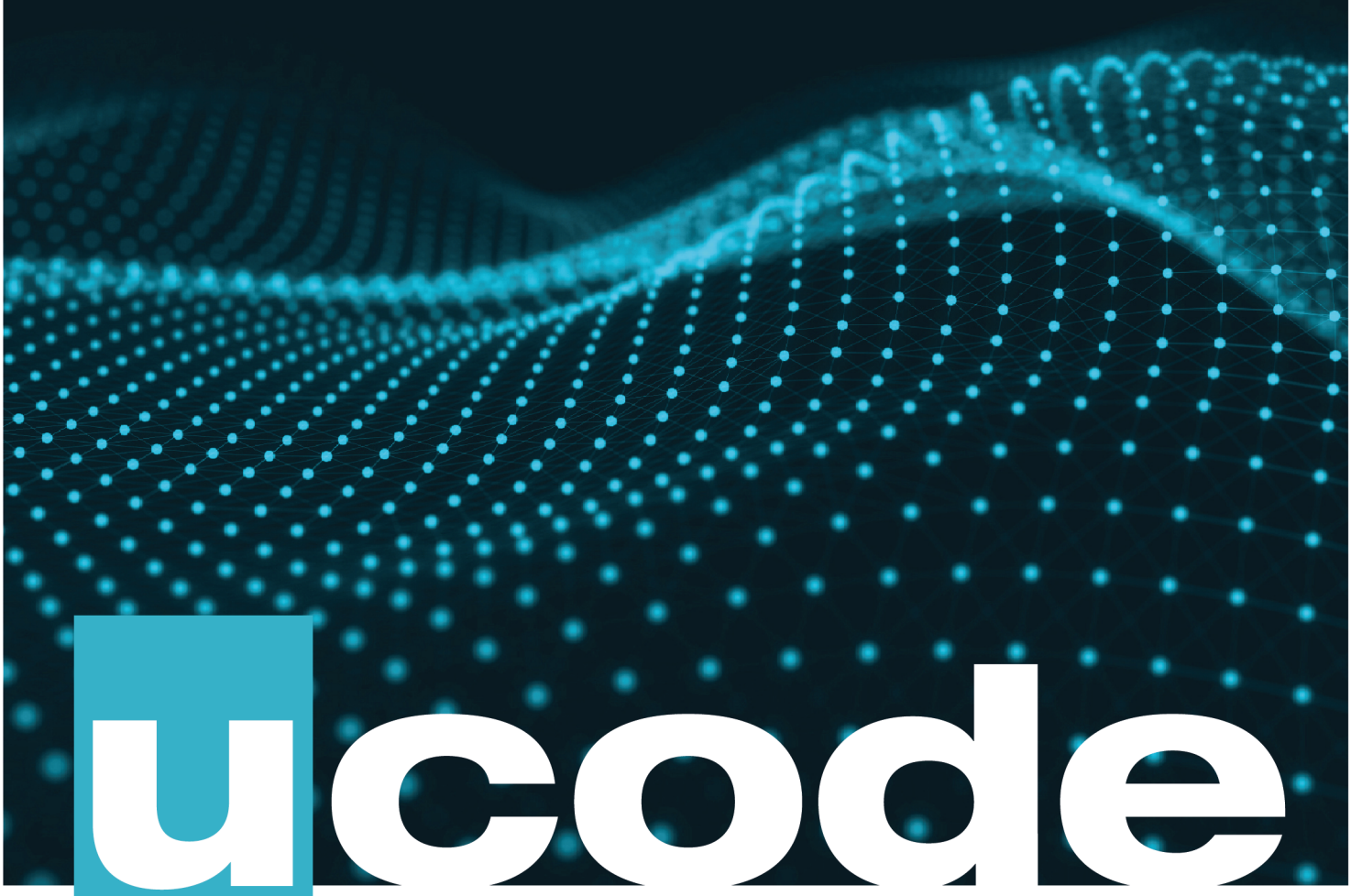
uchat

Track C

December 19, 2019



ucode

Contents

ucode

Engage

Final call!

Writing...Talking...Chatting...Messaging...
In the era of digitalization when Internet using has completely penetrated our lives, we have the ability to communicate with people around the world. We mostly use text messaging for this. And sending these messages takes seconds.

Each of us uses or has used messengers like Slack, Telegram, Viber, Snapchat and others. But at the same time, many people in the world are feeling a crisis of communication.

We invite you to make your contribution in bringing people together by putting yourself in the position of chat's and messengers' creators. At the same time you will get understanding of how electronic devices are exchanging the information on the basic level, without delving too deeply into the low-level nuances.

You will use Unix `sockets` to exchange data packages between server and client.

BIG IDEA

Solve communication problems of the world.

ESSENTIAL QUESTION

How to create user-friendly messaging with C.

CHALLENGE

Build cool socket chat using C.

Investigate

As you should know, sockets, which you must use here, plays a very important role in local and network processes communication.

They used in embedded systems, web-server engines, low-level system processes communication, etc.

The educational goal of this challenge is to create a multi-connected system using C and sockets API.

GUIDING QUESTIONS

We invite you to find answers to the following questions. This will help you and your team to realize which knowledge you will get from this challenge and how to move forward.

Ask your neighbor on the right, left, or behind you, and discuss the following questions together. You can find the answers in the Internet and share it with students around you.

We encourage you to find answers as many questions about developing chats as you want, the bigger, the better. These are some standart questions which are needed to be asked.

- What is IP address?
- How does works sockets in C?
- Why is text messaging so popular?
- What's your favourite messengers?
- What are the differences between process and thread?
- How to build a convenient interface for chat?
- Which functionality do people like in chats the most?
- Which frameworks can help you to create a GUI?
- What is UI/UX? Why it's important when you build the product for real users?

GUIDING ACTIVITIES

These are only a set of example activities and resources.

- Find out what is your working machine IP and what difference between IPv4 and IPv6.
- Find all information about `sockets` in `C`. Use them in practice.
- Read about libraries and frameworks that can help your team build GUI. For example read about GTK and SDL.
- Read about SQL, it's new language for you, simple but new.
- Try to understand how to organize working directory and write Makefile for this challenge. Remember that your Makefile must compile two executable files.
- Read about multithreading in C. Try to find a task where the use of threads is justified. `man pthread`.

ANALYSIS

You need to analyze all the collected information and create the concept of your solution.

- Discuss with your team the plan for implementing the chosen idea.
- Research the tools and technologies that will help you realize your solution.
- The challenge must be performed in `C`.
- Discuss your messaging app design. Create logic for how the user should use your product. In other words, make a brainstorm with your team and write down a detailed script of how the user will use your app.
- Do not stop on simple messaging. Make your chat like Pavel Durov but only better :)
- Think about the implementation of your application, think about important features. Be sure to pay attention to whether the user is interested in using your application.
- Do not forget, that the user interface should be easy to use.
- You should submit only files that you used to complete a challenge. Garbage shall not pass.
- Your program must manage memory allocations correctly. Memory which is no longer needed must be released otherwise the task is considered as incomplete.
- You must complete challenge according to the rules specified in `Auditor`.
- Your exercises will be checked and graded by students. The same as you. `Peer-to-Peer (P2P) learning`.
- Also, your exercises will pass automatic evaluation which is called `Oracle`.
- Got a question or you do not understand something? Ask the students or just Google that.
- Use your brain and follow the white rabbit to prove that you are the Chosen one!!!

Act

This challenge will introduce you with basics of the networks and how this working in C. To succeed you'll have to understand how to link all incoming connections. As you know, for now, a lot of different chat programs were created. So you have a good chance to understand how they work and how to make it better.
Most of the Net Applications use the Client-Server architecture, which refers to two processes or two applications that communicate with each other to exchange some information. One of the two processes acts as a client process, and another process acts as a server. So, the uchat is composed of two parts. First part - server. Server should handle requests from all client applications and must do it realy fast. The second part, as it easy to guess - client. Client, it's nothing more than a beautiful wrapper for the simple user, it should never make real work, only request to big brother (Server application) and nothing else.

Your goal? Write 2 programs in C:

- The first, named chat_server which takes an integer argument, network port. It should be running as the daemon. At startup, it'll display process id of this application.
- The second one is called uchat which takes two arguments, server IP address and port. It should establish the connection with the server application.

ALLOWED FUNCTIONS

- write, read
- malloc, calloc, free
- exit
- open, close
- socket, setsockopt, getsockopt, bind, listen, connect, accept, recv, send
- htons, htonl, inet_pton
- time, localtime, strftime
- All pthread library functionality
- perror, strerror
- fork
- setsid
- All ncurses library functionality
- All sqlite3 library functionality

Architecture

For imagine the basics of your future application, please decide what architecture you`ll implement. It gives a basis for analysis of software systems' behavior before the system has been built. Architecture helps to reduce risks and chance of failure. There are two types of client-server architectures:

- 2-tier architecture - in this architecture, the client directly interacts with the server. This type of architecture may have some security holes and performance problems. Internet Explorer and Web Server work on two-tier architecture. Here security problems are resolved using Secure Socket Layer (SSL).
- 3-tier architectures - in this architecture, one more software sits in between the client and the server. This middle software is called 'middleware'. Middleware are used to perform all the security checks and load balancing in case of heavy load. A middleware takes all requests from the client and after performing the required authentication, it passes that request to the server. Then the server does the required processing and sends the response back to the middleware and finally the middleware passes this response back to the client.

The server part

There are two types of servers you can have:

- Iterative Server - this is the simplest form of server where a server process serves one client and after completing the first request, it takes request from another client. Meanwhile, another client keeps waiting.
- Concurrent Servers - this type of server runs multiple concurrent processes to serve many requests at a time because one process may take longer and another client cannot wait for so long. The simplest way to write a concurrent server under Unix is to fork a child process to handle each client separately.
- You could write a program named chat_server, which will get the network port as an argument. If no argument are given chat_server stops and displays usage.
- chat_server should start as a daemon in a passive mode, listening to the specified port. Each connection should be correctly established. Maximum limit of the users is set only by the operational system.
- After the first client had been connected to chat_server, it should be correctly handled in an asynchronous mode without suspending all server.
- In case if the connection was aborted or some errors were found during this process, your server should correctly free all resources and continue work in the default way.
- If some client sends a message, it should be redirected to all connected clients except sender.
- When a client connects to / disconnects from the server, others should know it.
- It's good to know, that you don't lose the thread of conversation, so if the client connects to the server, all previous chat history should be available to him.
- The server should show minimal statistics, the number of users online or, for example, the number of chat rooms and other useful information for your choice.
- All history must be saved into the database, we advise you to use SQLite, but you can use simple representation like saving into json, yaml, xml or other file formats.

The client part

- You have to write a program named uchat which will receive as arguments the server IP address and port.
- The program must manage all line editing operations. If you are clever, you'll use readline library. This time, bicycle it's not a good idea.
- When the client receives a new message, it should be correctly displayed and prompt shouldn't be modified, even if you are writing something at this moment.
- If your uchat application lost connection with the server, you should manage reconnection. Appropriate messages should be displayed, also history shouldn't be loaded again, or displayed only new part.
- Create a simple and yet user-friendly client's graphic design. `ncurses` is a library you know, but it's time to build something bigger! We invite you to use any kind of graphic libraries that will not do all the work for you. But if you're not interested in that, you can still use `ncurses`. Using graphics libraries is an advanced part.
- Basic authentication (by username).
- The recipient must see the messages and the time when they were sent.

GUI

The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. GUI is a crucial way for computer-human interaction. The major benefit of a GUI is that systems using one are accessible to people of all levels of knowledge, from an absolute beginner to an advanced developer or other tech-savvy individuals. There are some design systems you could use in your project:

- Google Material Design;
- Flat Design;
- Apple Design;
- Fluent Design by Microsoft.

Usability

Usability is the ease of use and learnability of a developed software or human-made object such as a tool or device. The term 'usability' is connected with GUI. Your team could implement usability testing to develop the best application. Usability testing is a way to see how easy to use something is by testing it with real users. Usability testing can be used in a variety of ways during your project lifecycle. It has many advantages:

- feedback direct from the target audience to focus the project team;
- internal debates can be resolved by testing the issue to see how users react to the different options being discussed;
- issues and potential problems are highlighted before the product is launched.

Security

Client Server Security Architecture is
Making your applications to function correctly without addressing the security needs of your infrastructure could have devastating consequences down the line. There are a lot of ways to protect client-server architecture:

- Use SSL certificate on the server to ensure the traffic is secure.
- Have a firewall set up between the server and client.
- Have a separate database server.
- Store passwords in the database using a secure hashing function such as PBKDF2.
- User authentication.

The items outlined above are only some of the enhancements you can make to improve the security of your systems.

Act: Creative

ALLOWED FUNCTIONS

Any functions and libraries which using is justified
We invite you to create cool chat and implement all the features that you want. Remember, it's better to implement useful features, no need to add something that no one will use!
We wrote to your team a few examples:

- External graphics libraries for GUI creation.
- Encrypt all data which you sent.
- Advanced authentication (using username and password, the password must be stored in encrypted form).
- List of online users for client.
- Add private chats or channels to server engine. Let people be antisocial in a socialclub.
- Handle all incoming clients by the event-triggered system. It's mean, not to use slow threads.
- Detailed logging for server.
- The ability to change the design (theme, color, etc.) by the user, as an option to use a configuration file or setting page in the client program.
- Admin tools for server (block user for some time, kick user from the chat, moderate username, add to another room, delete messages, etc.).
- And more things limited only by your imagination.

Please do not delay the challenge for too long. Manage your time and time of your team carefully.

Share

Publishing

The final important and integral stage of your work is its publishing. This allows you to share your reflections and solutions on your challange with a local and global audience.

During this stage, you will find how to get a global assessment. You will get representative feedback. As a result, you get the maximum experience from the work you have done. Do not forget to share your teamwork experience.

What you can create to disseminate information

- Text post, summary of reflections.
- Charts, infographics or any other ways to visualize your information.
- Video of your work.
- Audio podcast. You can record a story with your experience.
- Photos from ucode with small post.

Example techniques

- Canva - a good way to visualize your data.
- QuickTime - easy way to record your screen, capture video, or record audio.

Example ways to share your experience

- Facebook - create a post that will inspire your friends.
- YouTube - upload a video.
- GitHub - share your solution.
- Telegraph - create a post. This is a good way to share information in a Telegram.
- Instagram - share a photos and stories from ucode. Don't forget to tag us :)

Share what you learned with your local community and the world. Use #ucode and #CBLWorld on social media.