

Tutorial: C-Chat

Rafael Renó Corrêa, Lucas Ferreira Alves, Gabriel da Silva Henrique

Junho de 2025

Sumário

Sumário	2	
0.1	Introdução	3
0.1.1	Referencial Teórico	3
0.1.2	Arquitetura	3
0.2	Tutorial	6
0.2.1	Requisitos	6
0.2.2	Compilação	6
0.2.3	Execução	6
0.2.3.1	Servidor	6
0.2.3.2	Cliente	6
0.2.4	Utilização	7
0.3	Compatibilidade	9

0.1 Introdução

A comunicação representa um dos pilares da vida social humana, onde o desenvolvimento da tecnologia propiciou formas de comunicação cada vez mais rápidas e eficientes. Nesse cenário, desde a invenção do telégrafo até a consolidação da *internet* por fibra-ótica, os aspectos do dia-a-dia estão cada vez mais voltados para as diferentes mídias sociais e ferramentas de comunicação pela rede de computadores, como Whatsapp, Messenger, Telegram, etc.

Entretanto, apesar da tendência atual de crescimento e acessibilidade dessas formas de comunicação, a maioria desses serviços de comunicação interpessoal por computador pertence a um grupo seletivo de organizações que, muitas vezes, utilizam dos dados de usuário para instrumentalizar estratégias de distribuição de anúncios ou manipulação política. É nesse contexto que a ferramenta C-Chat pode ser útil, uma vez que implementa uma arquitetura de comunicação sem armazenamento de qualquer tipo de informação de usuário em longo prazo, através de uma ferramenta sofisticada e eficiente, implementada em C.

0.1.1 Referencial Teórico

Nesse cenário, as tecnologias utilizadas foram o arquivo de cabeçalho da biblioteca padrão de uso geral na linguagem de programação C (`stdlib.h`) e o arquivo de cabeçalho que fornece acesso a funções da API do sistema operacional POSIX em C (`unistd.h`) – para manipulação de processos. Além disso, para a manipulação de *threads*, foram utilizadas diretivas definidas pela biblioteca OpenMP (`omp.h`). A comunicação entre processos no mesmo grupo foi sincronizada pela implementação das bibliotecas de semáforo do (`sys/sem.h`), espaço de memória compartilhado (`sys/shm.h`), comunicação entre processos (`sys/ipc.h`) e tratamento de sinais (`signal.h`). Por fim, todo tipo de atividade em camada de rede foi implementada utilizando comunicação orientada à conexão (IPv4 e TCP/IP) por soquetes de rede (`sys/socket.h`). Além de outras bibliotecas utilizadas para modularização de tarefas e simplificação do código, tais como: `stdio.h`, `arpa/inet.h`, `netinet/in.h`, `string.h`, `wait.h` e `errno.h`.

Para a interface gráfica, foi implantada a biblioteca *DearImGui*, uma biblioteca leve de interface gráfica para C++, projetada para facilitar o desenvolvimento de interfaces dinâmicas e depuração em tempo real. Ela segue o paradigma *immediate mode GUI*, no qual a interface é reconstruída a cada *frame* com chamadas diretas de código, eliminando a necessidade de armazenar ou gerenciar o estado da interface entre *frames*. O *ImGui* não faz a renderização por conta própria — em vez disso, ele gera *vertex buffers* e listas de comandos de desenho otimizadas. Onde, especificamente neste projeto, são renderizados utilizando a API gráfica *OpenGL*, enquanto a criação da janela e o gerenciamento de entradas (como teclado e mouse) são realizados pela biblioteca *GLFW*.

0.1.2 Arquitetura

O C-Chat funciona com uma arquitetura simples no modelo cliente-servidor, onde uma máquina servidora principal orquestra a comunicação de múltiplos clientes de forma concorrente. Para isso, implementa multiprocessamento e *multithreading* com uma estrutura robusta: No processo pai da aplicação servidora, o programa estabelece novas conexões com máquinas clientes. Uma vez estabelecida, o código é herdado por um processo filho que manipula a comunicação de entrada e saída de dados com o cliente utilizando duas *threads* concorrentes. Enquanto isso, o processo pai se prepara para estabelecer novas conexões.

Dessa forma, o processo filho que gerencia o cliente atua como a interface de aplicação do servidor para o cliente específico. Onde a *thread* de entrada recebe mensagens do cliente e a *thread* de entrada envia

mensagens de outros membros do grupo ao cliente. Para isso, processos filhos diferentes pertencentes ao mesmo grupo compartilham um espaço de memória compartilhado onde, um por vez, escreve ou lê o *buffer* compartilhado. Assim, para enviar a mensagem do cliente recebida para os outros membros do grupo, a *thread* de entrada do cliente escreve a mensagem na memória compartilhada, posteriormente, a mensagem é lida pela *thread* de saída dos outros membros do grupo que a encaminham para os respectivos clientes.

Note que, uma vez que todas as *threads* são concorrentes, foi necessário implementar um sistema de semáforo para impedir o acesso simultâneo à memória compartilhada – pois somente um processo pode ler ou escrever no mesmo espaço de memória ao mesmo tempo – além disso, para evitar *starvation*, onde um processo sobrescreve a memória compartilhada antes de todos os outros membros do grupo lerem a mensagem, foi implementado um contador na mensagem: Se houver X membros no grupo, o processo só sobrescreve o *buffer* compartilhado se o contador da mensagem for exatamente X, se não, aguarda e tenta novamente após 250 milissegundos. Portanto, segue o diagrama em alto nível dessa arquitetura na Figura 1:

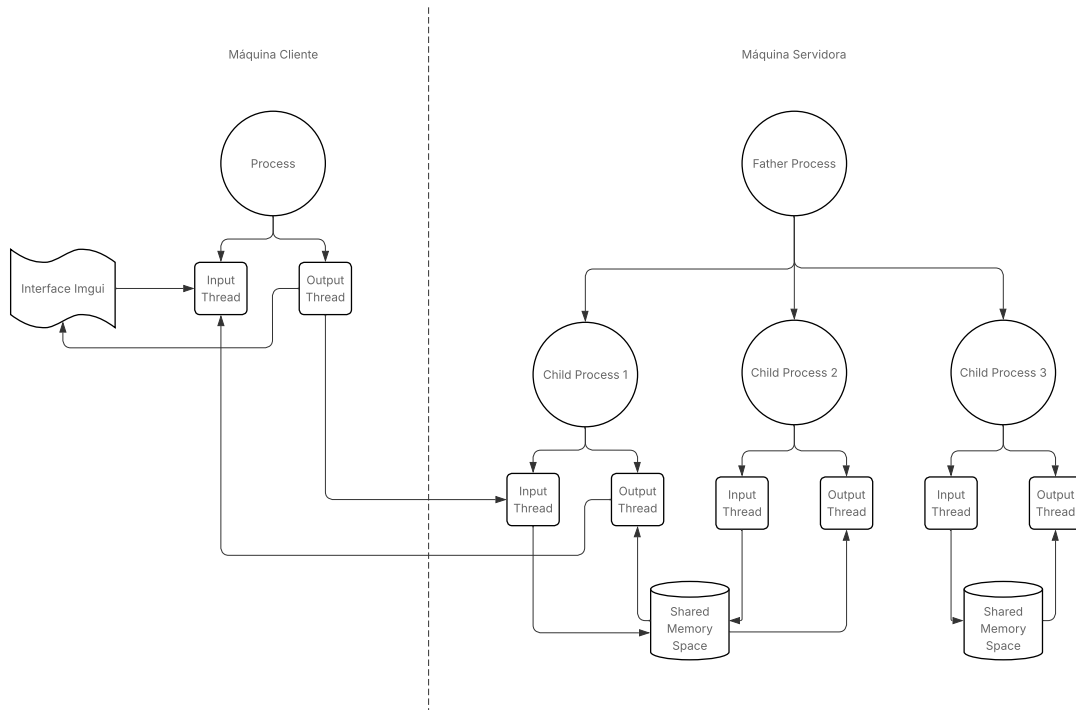


Figura 1 – Arquitetura do C-Chat.

Note que, dada a natureza do projeto, existem duas interfaces disponíveis: Por linha de comando (CLI) e ImGui (GUI). Assim sendo, a aplicação cliente pode funcionar tanto como interface ou intermediador (*middleware* mensageiro). Assim sendo, se for interface CLI, a aplicação cliente estabelece conexão com o servidor e inicia duas *threads* concorrentes para leitura e escrita (pelo terminal). Se for interface ImGui, a aplicação cliente estabelece conexão com a aplicação da interface e inicia as duas *threads* que, desta vez, recebe as mensagens da interface e as encaminha para o servidor, e vice-versa. Isto é, como está disponível no fluxograma na Figura 2:

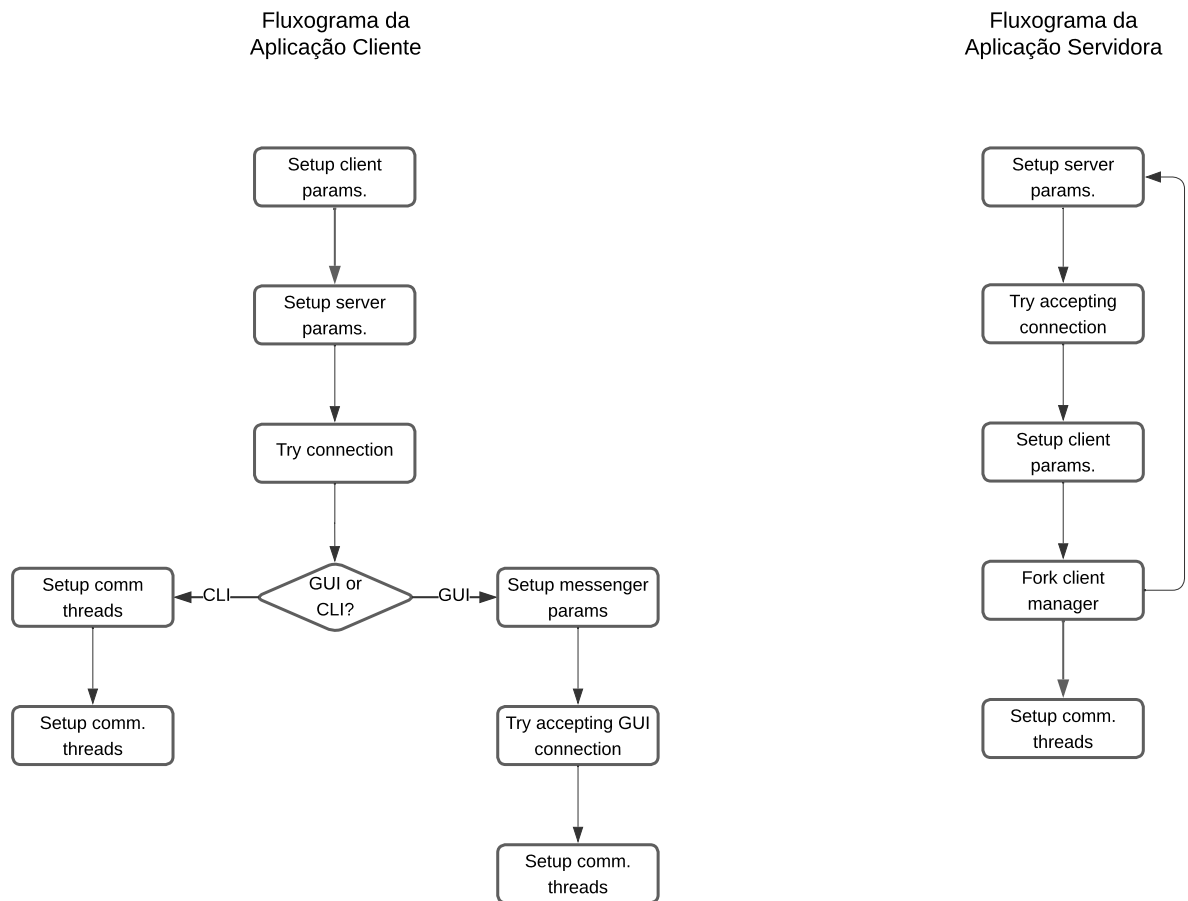


Figura 2 – Fluxograma de ações do C-Chat.

0.2 Tutorial

Finalizando, para compilar, executar e utilizar a aplicação, um passo-a-passo foi dividido em quatro subseções: Requisitos, Compilação, Execução e Utilização.

0.2.1 Requisitos

A interface gráfica implementa OpenGL + GLFW. Dessa forma, pode ser necessário instalar o GLFW em algumas distribuições Linux:

sudo pacman -S glfw (Arch) ou **sudo apt install libglfw3** (Ubuntu).

Além disso, para compilar os executáveis, é necessário instalar o GCC e CMake:

sudo pacman -S gcc make (Arch) ou **sudo apt install gcc make** (Ubuntu).

0.2.2 Compilação

No diretório "core", execute as seguintes instruções de compilação das aplicações cliente e servidora:

gcc -c server_utils.c -o server_utils.o

gcc -c client_utils.c -o client_utils.o

gcc server.c client_utils.o server_utils.o -o server -fopenmp

gcc client.c client_utils.o -o client -fopenmp

Por fim, para compilar a interface gráfica baseada em ImGui, basta executar **make** no diretório "src".

0.2.3 Execução

0.2.3.1 Servidor

Em uma máquina, execute o arquivo do servidor compilado (**server**) em "core" com a porta escolhida no segundo parâmetro, ou seja,

./server 8080 para iniciar o servidor na porta 8080.

Certifique-se que o *firewall* da máquina está liberado para entrada e saída de dados na porta 8080 (ou na porta escolhida): **sudo ufw allow 8080/tcp**.

0.2.3.2 Cliente

Em **/core/client.c**, altere a definição **SERVER_IP** (linha 35) para o endereço IPv4 da máquina servidora, posteriormente, execute a aplicação com os parâmetros de "nome de usuário", "grupo" e "porta". Por exemplo,

./client Rafael 123 8080 para iniciar o cliente na porta 8080, com nome de usuário "Rafael" e no grupo "123". Dessa forma, estará comunicando com qualquer outros clientes no grupo "123".

Se preferir a interface gráfica, antes de compilá-la, altere a definição **PORT** (linha 55) em **/src/main.cpp** para a porta escolhida na máquina servidora (8080, por exemplo). Em seguida, compile e execute a aplicação através da instrução **./client_gui** em "src".

0.2.4 Utilização

Logo após iniciar a interface gráfica, haverá uma única janela com dois campos de entrada (Figura 3):

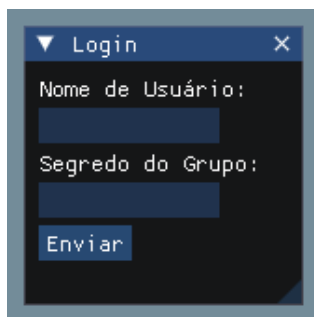
A screenshot of a small window titled "Login" with a close button (X) in the top right corner. Inside the window, there are two text input fields. The first field is labeled "Nome de Usuário:" and the second is labeled "Segredo do Grupo:". Below these fields is a blue button labeled "Enviar".

Figura 3 – Formulário de *Login*.

Preencha com o nome de usuário e código do grupo desejado, em seguida, pressione "Enviar". Em instantes, outra janela aparecerá, referente ao grupo do código inserido (Figura 4):

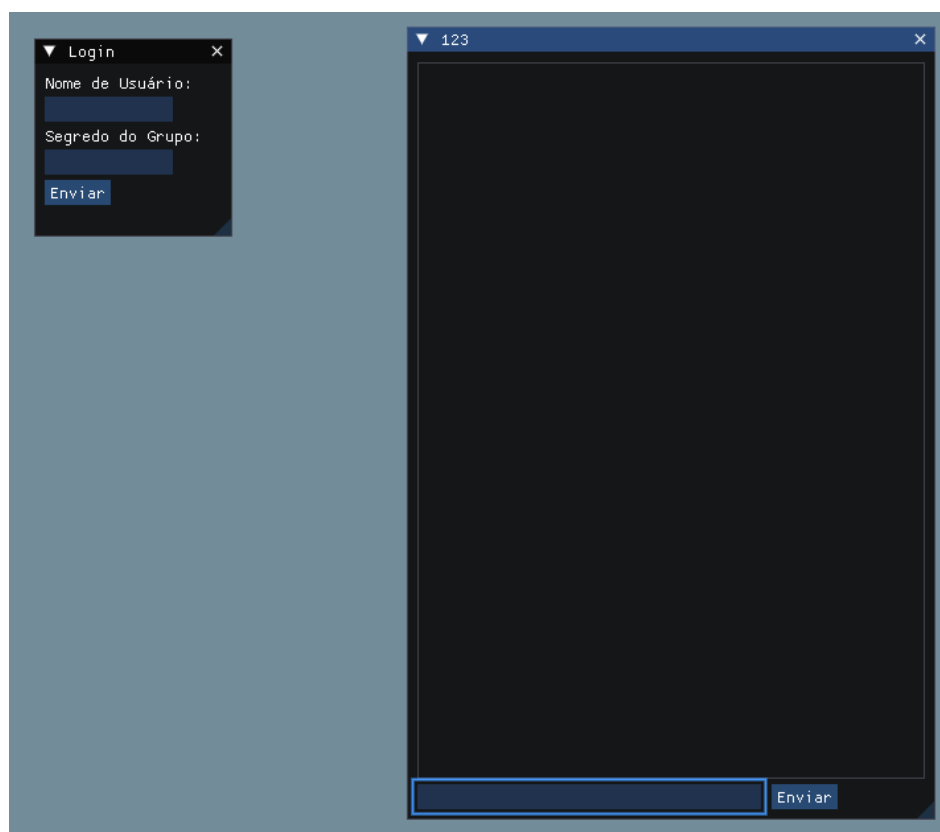
A screenshot of the main application window. On the left, the "Login" window from Figure 3 is still visible. On the right, a larger window titled "123" is open, representing a chatroom. This window has a large black area for messages and a text input field at the bottom with a blue "Enviar" button.

Figura 4 – *Chatroom* aberta.

Para enviar uma mensagem, basta preencher o campo de texto e pressionar a tecla "Enter" ou clicar em "Enviar" (Figura 5):

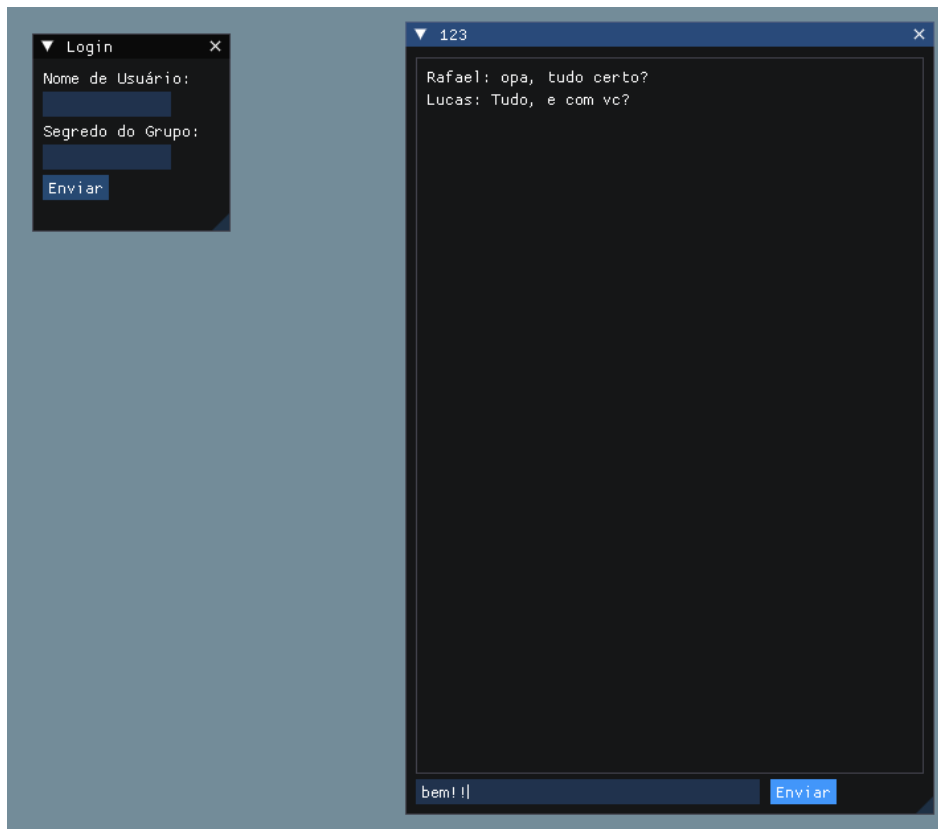


Figura 5 – Exemplo de envio de mensagem.

Se desejar, é possível entrar em vários grupos ao mesmo tempo! Basta inserir o nome de usuário desejado e o código do grupo (Figura 6):

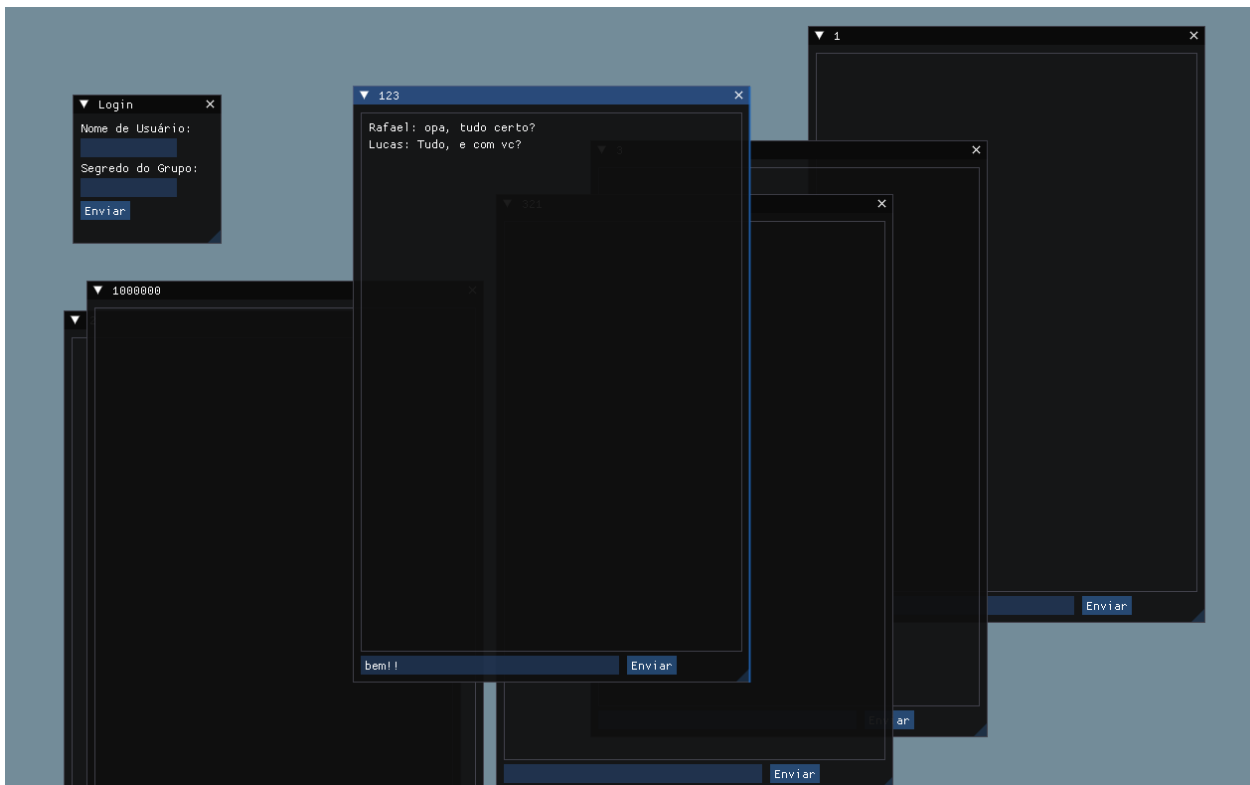


Figura 6 – Exemplo de várias *chatrooms* simultâneas.

Note que, por escolha de *design*, não é possível entrar no mesmo grupo em duas instâncias de *chat* no mesmo computador. Além disso, o envio de mensagens repetidas não é encaminhado para os outros membros do grupo a partir da segunda mensagem repetida enviada.

0.3 Compatibilidade

A aplicação está disponível para sistemas Linux. Testado no Arch Linux x86_64.