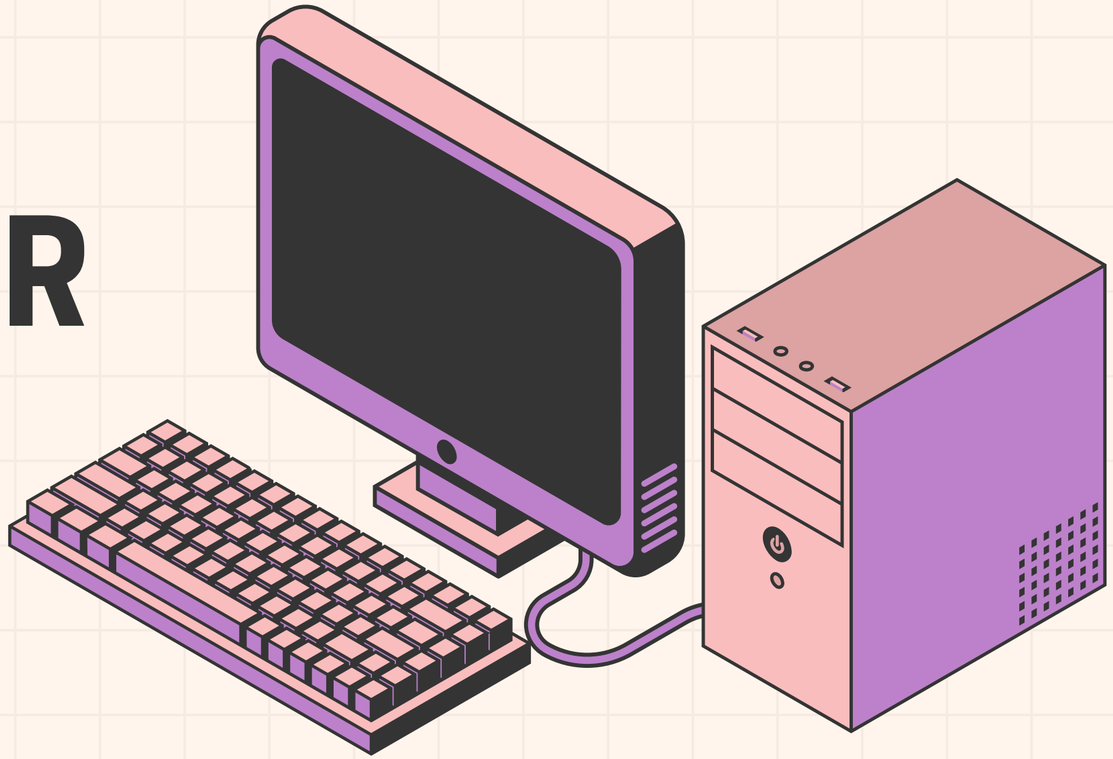
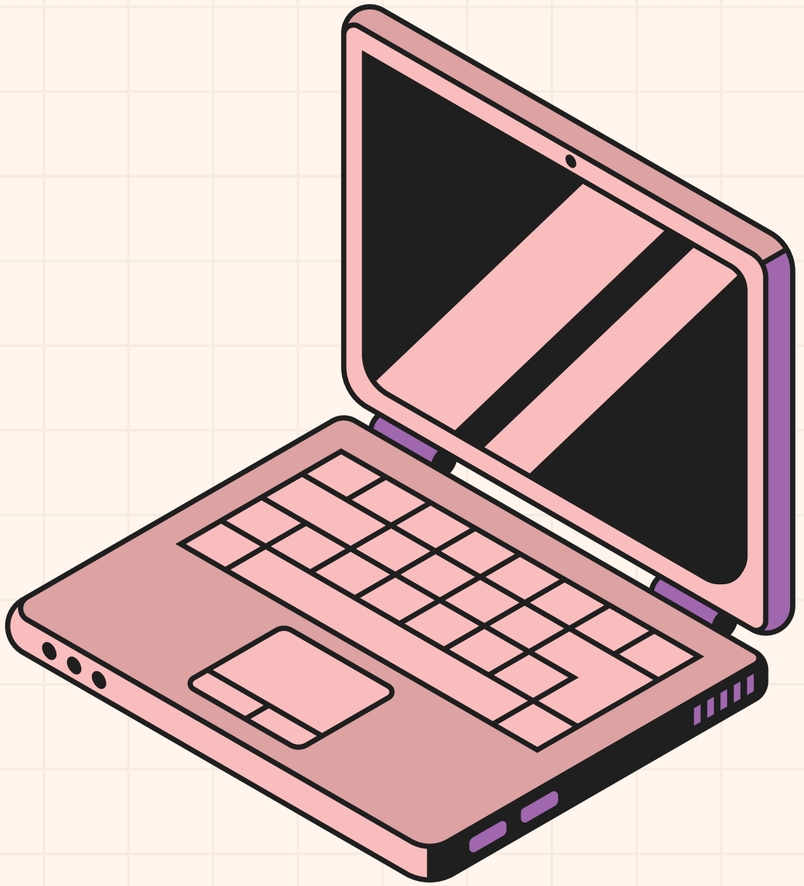
CLIENTE/SERVIDOR SEGURO

Alunos:
Guilherme Miranda de Araújo
Hendrick Silva Ferreira
Vitor Jordão Carneiro Briglia



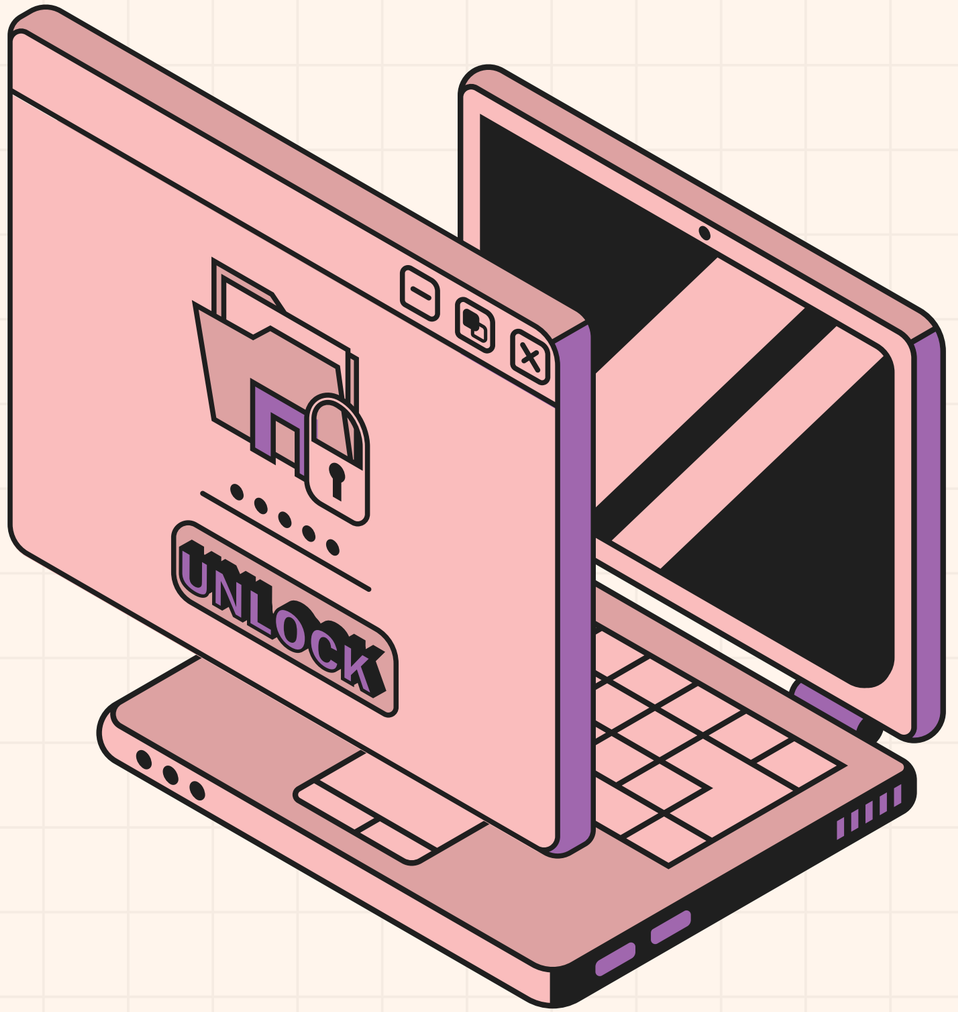
OBJETIVO

- Trabalho baseado na arquitetura do Telnet.
- Desenvolver um sistema de comunicação cliente-servidor seguro e moderno.
- Tecnologias utilizadas:
  - TCP/IP com sockets
  - Compressão de dados
  - Suporte a múltiplos clientes
  - Inspiração no modelo de terminal remoto



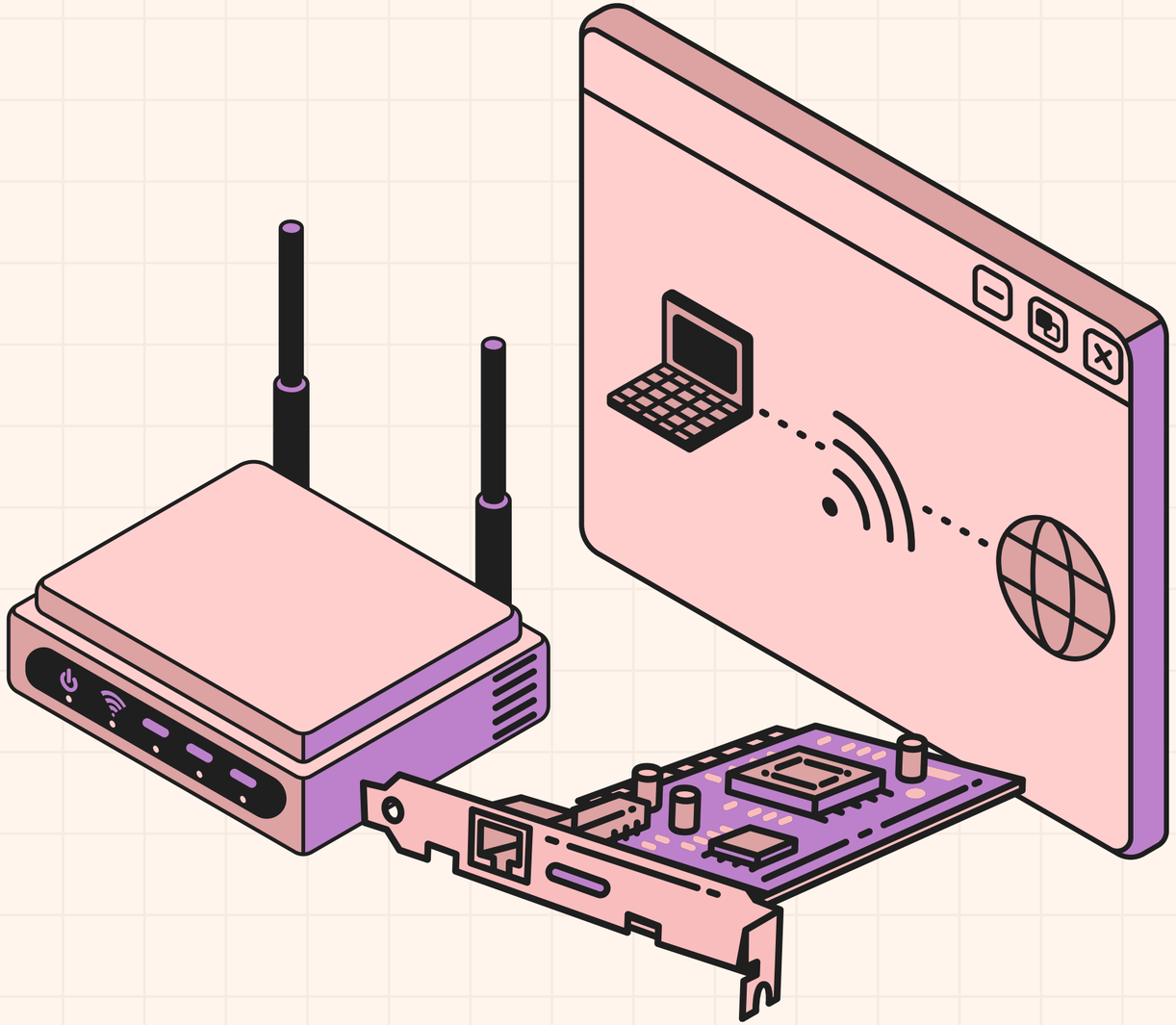
O QUE É TELNET?

- Telnet (TELecommunication NETwork) é um protocolo de rede criado nos anos 1970.
- Permite o acesso remoto a computadores através de uma linha de comando.
- Com Telnet, um usuário pode se conectar a um servidor remoto e executar comandos como se estivesse localmente no terminal.
- Utiliza o protocolo TCP na porta padrão 23.
- Principal limitação: não oferece criptografia — todos os dados (incluindo senhas) trafegam em texto puro.
- Foi amplamente substituído por protocolos mais seguros, como o SSH.



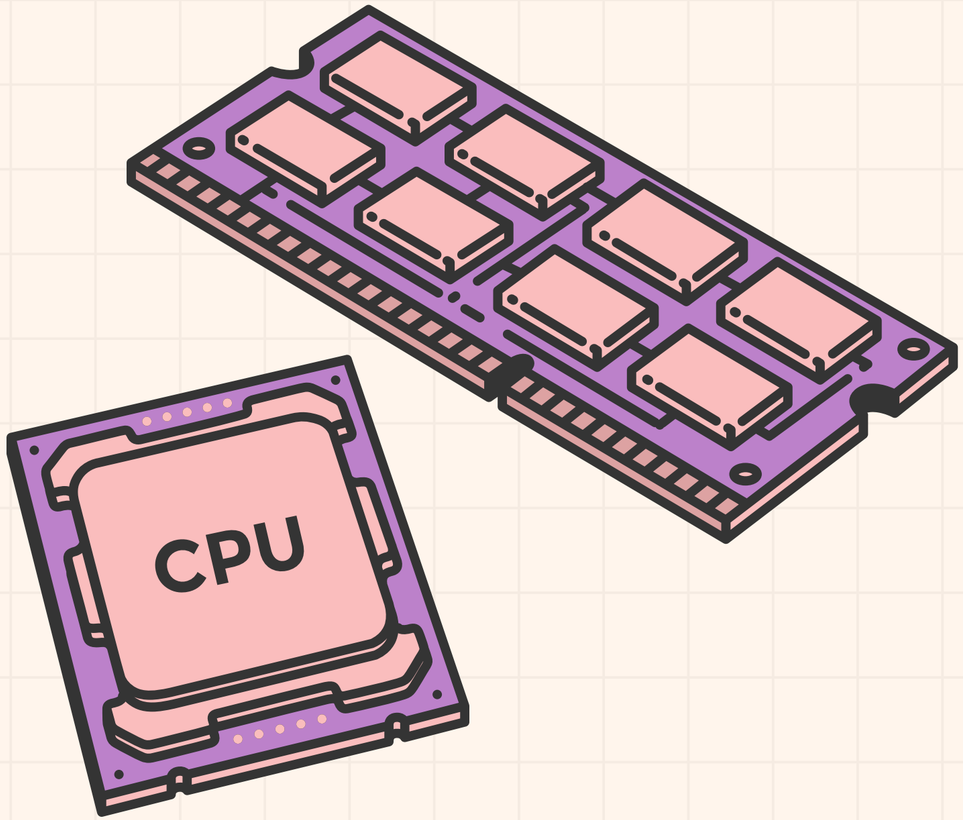
COMUNICAÇÃO
CLIENTE-SERVIDOR

- TCP/IP comparado a uma ligação telefônica: cliente “chama”, servidor “atende”.
- Cliente precisa do IP e da porta do servidor.
- Comunicação bidirecional após conexão.
- Confiabilidade: verificação de integridade dos dados.
- No projeto: uso de sockets TCP e testes via localhost.



PROCESSOS
FORKS & DAEMONS

- Forks: Criação de novos processos para atender múltiplos clientes simultaneamente.
  - Cada cliente tratado de forma isolada melhora a escalabilidade.
  - Ambiente mais robusto e multitarefa.
- Daemons: processos que rodam em segundo plano.
  - Independentes do terminal, utilizados como serviços permanentes.
  - Abordam conceitos modernos de gerenciamento de servidores.



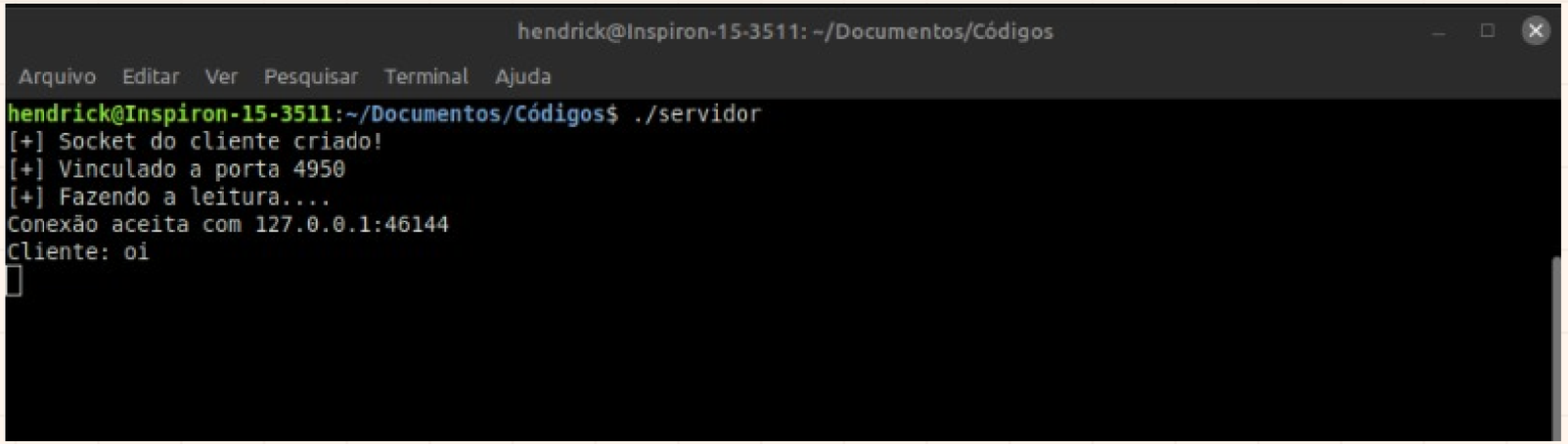
CÓDIGO
CLIENTE

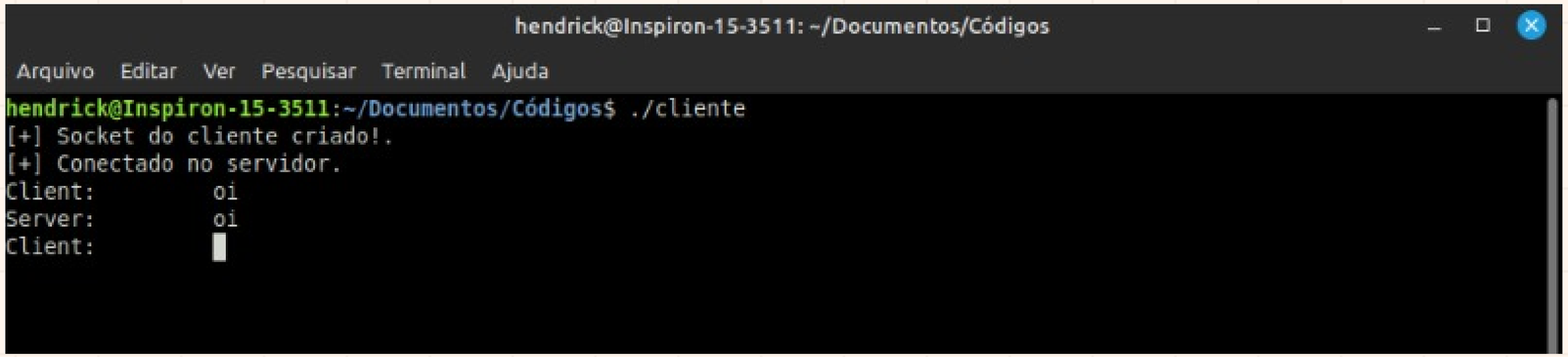
```c

int main(){
    int clientSocket, ret;
    struct sockaddr_in serverAddr;
    char buffer[1024];

    // Criação do socket
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if(clientSocket < 0){
        printf("[-] Erro na criação do socket.\n");
        exit(1);
    }
    printf("[+] Socket do cliente criado!\n");

    // Configuração do endereço do servidor
    memset(&serverAddr, '\0', sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Tentativa de conexão
    ret = connect(clientSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
    if(ret < 0){
        printf("[-] Erro na conexão.\n");
        exit(1);
    }
    printf("[+] Conectado ao servidor.\n");

    while(1){
        bzero(buffer, sizeof(buffer));
        printf("Client:\t");
        scanf("%s", buffer);  // cuidado: não lê espaços!

        send(clientSocket, buffer, strlen(buffer), 0);

        if(strcmp(buffer, ":exit") == 0){
            close(clientSocket);
            printf("[-] Conexão encerrada com o servidor.\n");
            break;
        }

        bzero(buffer, sizeof(buffer));
        if(recv(clientSocket, buffer, sizeof(buffer), 0) < 0){
            printf("[-] Erro ao receber dados.\n");
        }else{
            printf("Server:\t%s\n", buffer);
        }
    }

    return 0;
}


```

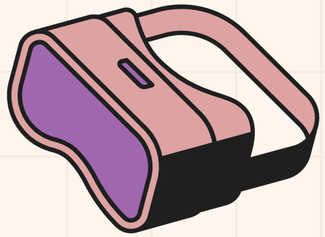
CÓDIGO
SERVIDOR

```c

    int newSocket;
    struct sockaddr_in newAddr;
    socklen_t addr_size;

    char buffer[1024];
    pid_t childpid;

    // Criação do socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd < 0){
        printf("[-] Erro na criação do socket.\n");
        exit(1);
    }
    printf("[+] Socket do servidor criado!\n");

    // Permitir reuso da porta
    int opt = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

    // Configuração do endereço
    memset(&serverAddr, '\0', sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Associação do socket com o endereço local
    ret = bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
    if(ret < 0){
        printf("[-] Erro na vinculação. \n");
        exit(1);
    }
    printf("[+] Vinculado à porta %d\n", PORT);

    if(listen(sockfd, 10) == 0){
        printf("[+] Aguardando conexão...\n");
    }else{
        printf("[-] Erro ao escutar.\n");
    }

    while(1){
        addr_size = sizeof(newAddr);
        newSocket = accept(sockfd, (struct sockaddr*)&newAddr, &addr_size);
        if(newSocket < 0){
            exit(1);
        }
        printf("Conexão aceita com %s:%d\n", inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));

        if((childpid = fork()) == 0){
            close(sockfd);

            while(1){
                bzero(buffer, sizeof(buffer));
                recv(newSocket, buffer, sizeof(buffer), 0);
                if(strcmp(buffer, ":exit") == 0){
                    printf("Desconectado de %s:%d\n", inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));
                    break;
                }else{
```

CÓDIGOS
RODANDO







OBRIGADO!