

COMPRESSED NETWORK COMMUNICATION

Sistemas
Operacionais
UFRR



INTRODUÇÃO

SERVIDOR TELNET

O Telnet existe há mais de 40 anos, muito antes de aparecer a Internet. Este sistema de transmissão de dados foi inventado pelas Forças Armadas

Americanas para transmissão de dados entre bases militares. Foi disponibilizado ao público em 1977, tendo sido os radioamadores os primeiros a aproveitá-lo.

O Telnet é um protocolo que permite emular um terminal à distância, isto é, que permite executar comandos escritos no teclado de um computador remoto.. Com Telnet podemos fazer "login" em outros computadores da Internet e utilizar os seus recursos. Por exemplo, executar aplicações e/ou aceder a serviços existentes nos computadores remotos. O nosso computador passa a ser como que um terminal (não inteligente) directamente ligado ao computador remoto (onde é executado todo o processamento).



TCP/IP



E protocolo TCP/IP nada mais é que um grupo de protocolos de comunicação cujo objetivo é entregar pacotes de dados entre dispositivos de origem e destino, usando informações de endereço.

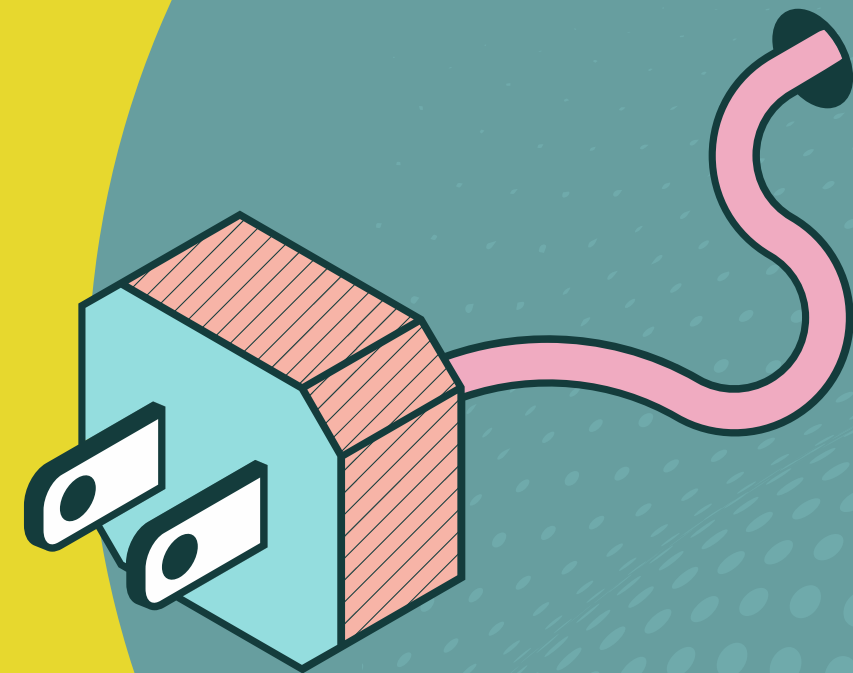
De uma forma simples, o TCP/IP é o principal protocolo de envio e recebimento de dados MS internet. TCP significa Transmission Control Protocol (Protocolo de Controle de Transmissão) e o IP, Internet Protocol (Protocolo de Internet).

Na realidade, o TCP/IP é um conjunto de protocolos. Esse grupo é dividido em quatro camadas: aplicação, transporte, rede e interface. Cada uma delas é responsável pela execução de tarefas distintas. Essa divisão em camadas é uma forma de garantir a integridade dos dados que trafegam pela rede.

TCP/IP

O CÓDIGO

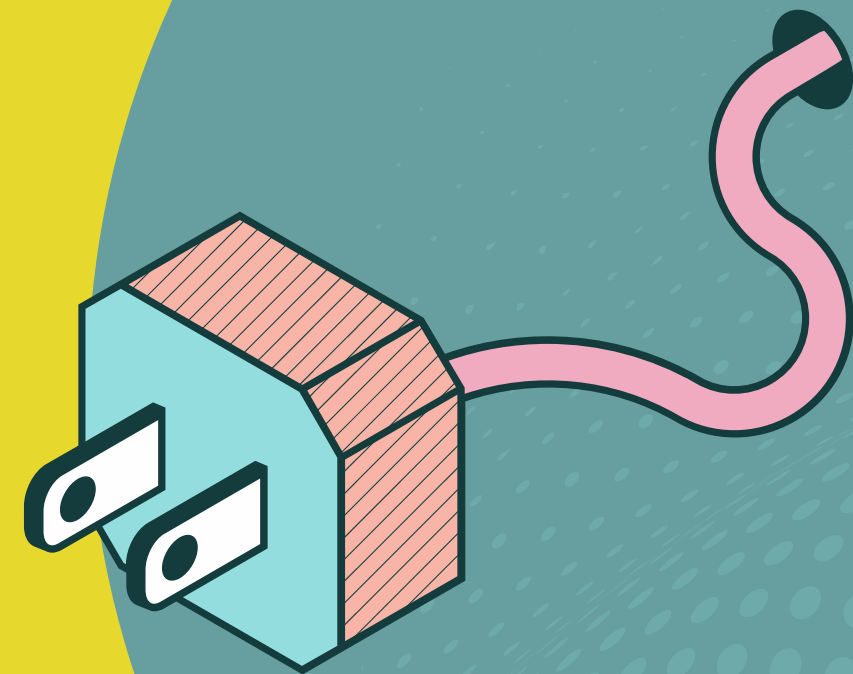
No caso, no nosso projeto, o TCP foi usado para fazer a comunicação entre os nossos códigos de “Cliente” e “Servidor”, e com o auxílio também do Socket, que é usado para troca de informações entre processos na mesma máquina ou em uma rede, distribuem o trabalho para a máquina mais eficiente e permitem o acesso fácil a dados centralizados, pois o nosso “Cliente” e “Servidor” estão sendo executados por via de localhost.



SOCKET

O CÓDIGO

Socket provê a comunicação entre duas pontas (fonte e destino) - também conhecido como two-way communication - entre dois processos que estejam na mesma máquina (Unix Socket) ou na rede (TCP/IP Sockets). Na rede, a representação de um socket se dá por ip:porta, por exemplo: 127.0.0.1:4477 (IPv4). Um socket que usa rede é um Socket TCP/IP.



FORK

A função fork é uma função que duplica o processo atual dentro do sistema operacional. O processo que inicialmente chamou a função fork é chamado de processo pai. O novo processo criado pela função fork é chamado de processo filho. Todas as áreas do processo são duplicadas dentro do sistema operacional (código, dados, pilha, memória dinâmica).

Caso a função fork retorne 0 (zero), está se executando o processo filho. Caso a função retorne um valor diferente de 0 (zero), mas positivo, o processo pai está sendo executado. O valor retornado representa o PID do processo filho criado. A função retorna -1 em caso de erro (provavelmente devido a se ter atingido o limite máximo de processos por usuário configurado no sistema).



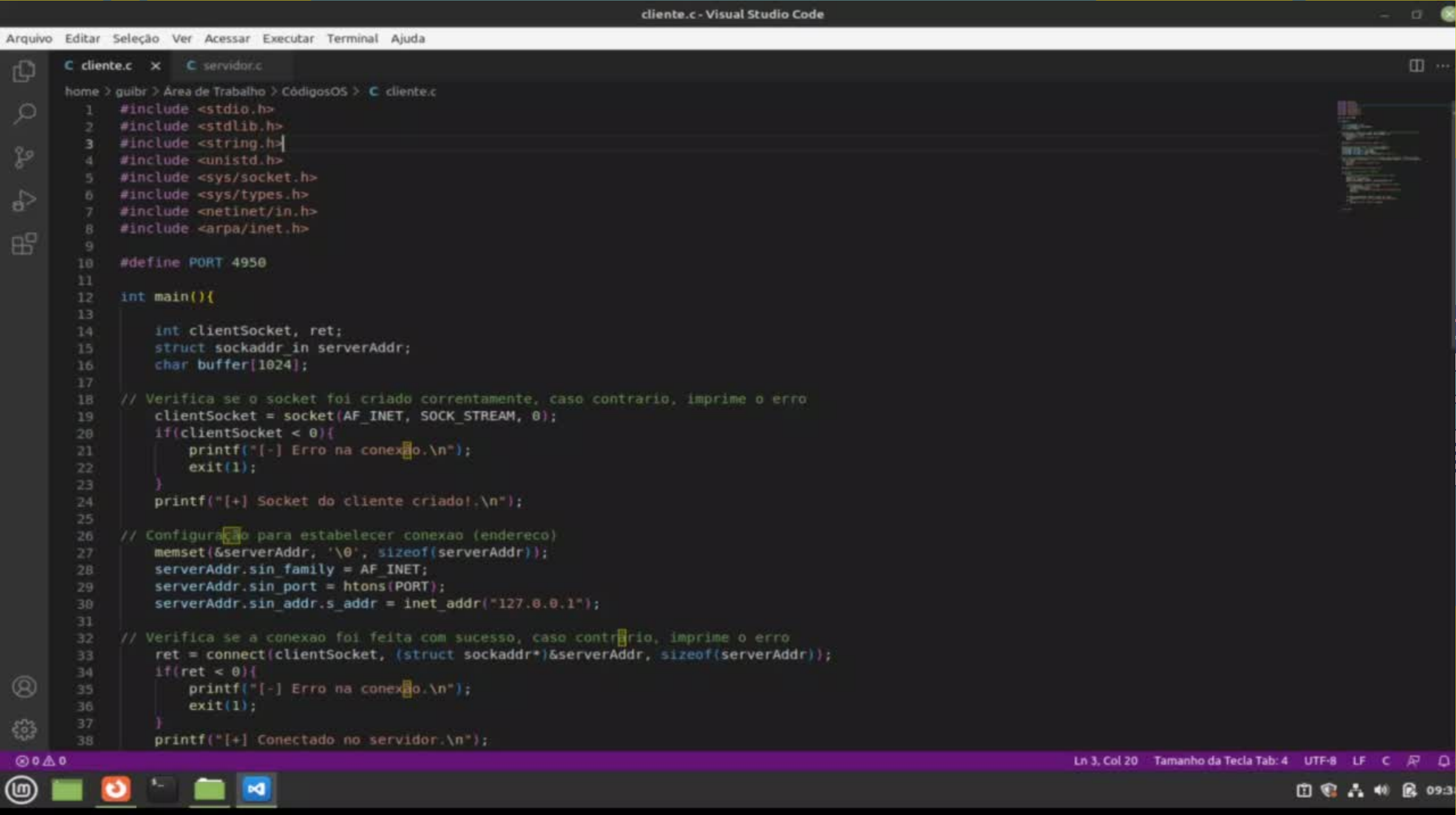
DESENVOLVIMENTO

O CÓDIGO

Para o desenvolver os códigos do “Cliente” e “Servidor”, nós usamos algumas bibliotecas como por exemplo `<sys/socket.h>`, para uso do Socket. Durante a criação do código tivemos alguns desafios em fazer com que o “Servidor” mandasse as mensagens para o “Cliente”, mas então conseguimos achar uma solução na criação de um “Echo Server”, que é um “Servidor” que envia de volta a mesma mensagem que o “Cliente” enviou ao “Servidor”. E como queríamos um “Servidor” que conseguisse atender multiclientes e se comunicasse com uma shell via pipe, decidimos utilizar a ajuda do Fork, que é criar um novo processo, que se torna o processo filho do chamador. Após a criação de um novo processo filho, ambos os processos executarão a próxima instrução seguindo a chamada de sistema `fork()`.



CLIENTE

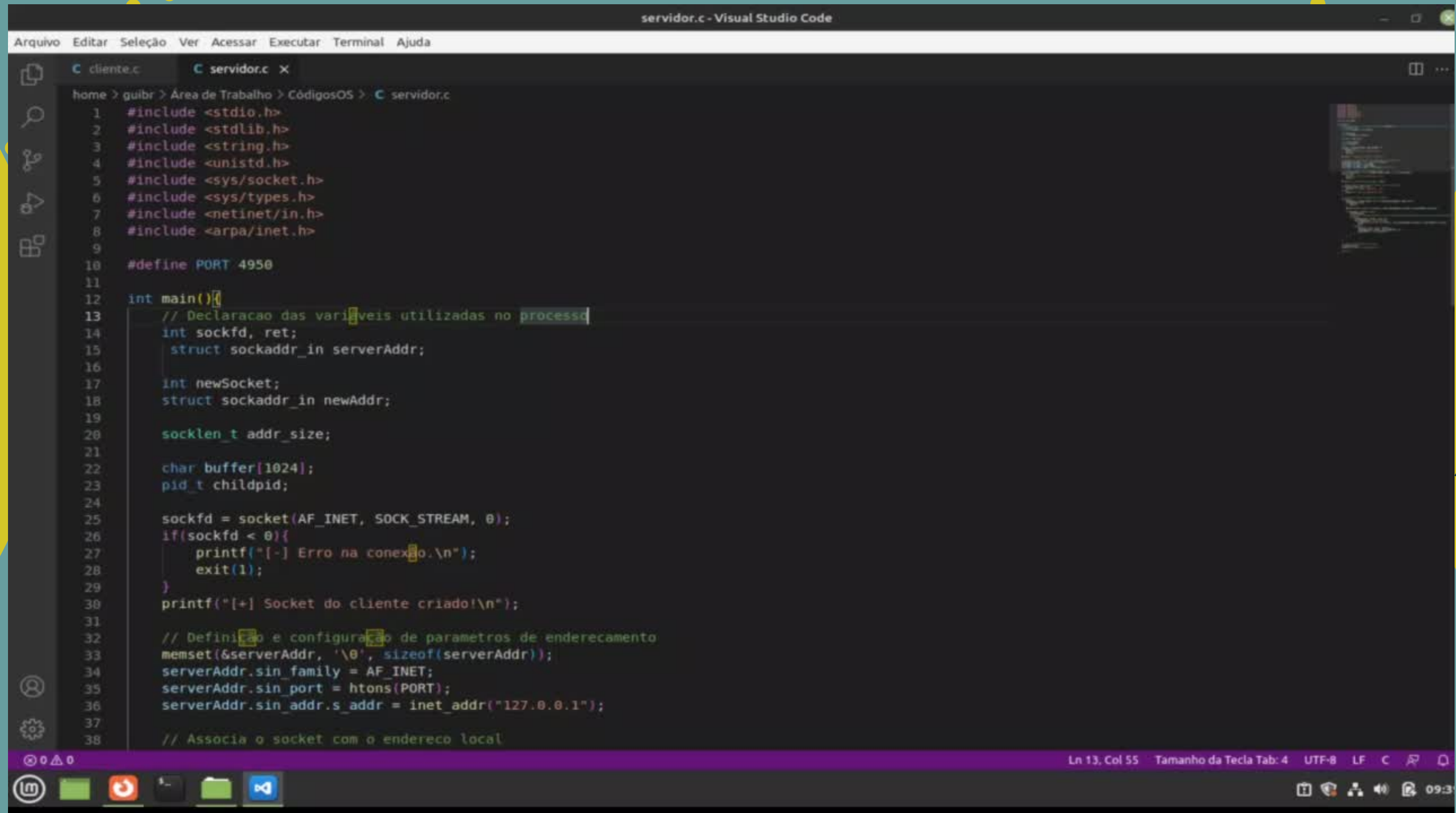


```
cliente.c - Visual Studio Code
Arquivo  Editar  Seleção  Ver  Acessar  Executar  Terminal  Ajuda

C cliente.c  x  C servidor.c
home > guibr > Área de Trabalho > CódigosOS > C cliente.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/socket.h>
6  #include <sys/types.h>
7  #include <netinet/in.h>
8  #include <arpa/inet.h>
9
10 #define PORT 4950
11
12 int main(){
13
14     int clientSocket, ret;
15     struct sockaddr_in serverAddr;
16     char buffer[1024];
17
18     // Verifica se o socket foi criado corretamente, caso contrário, imprime o erro
19     clientSocket = socket(AF_INET, SOCK_STREAM, 0);
20     if(clientSocket < 0){
21         printf("[-] Erro na conexão.\n");
22         exit(1);
23     }
24     printf("[+] Socket do cliente criado!\n");
25
26     // Configuração para estabelecer conexão (endereço)
27     memset(&serverAddr, '\0', sizeof(serverAddr));
28     serverAddr.sin_family = AF_INET;
29     serverAddr.sin_port = htons(PORT);
30     serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
31
32     // Verifica se a conexão foi feita com sucesso, caso contrário, imprime o erro
33     ret = connect(clientSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
34     if(ret < 0){
35         printf("[-] Erro na conexão.\n");
36         exit(1);
37     }
38     printf("[+] Conectado no servidor.\n");
```


SERVIDOR

8



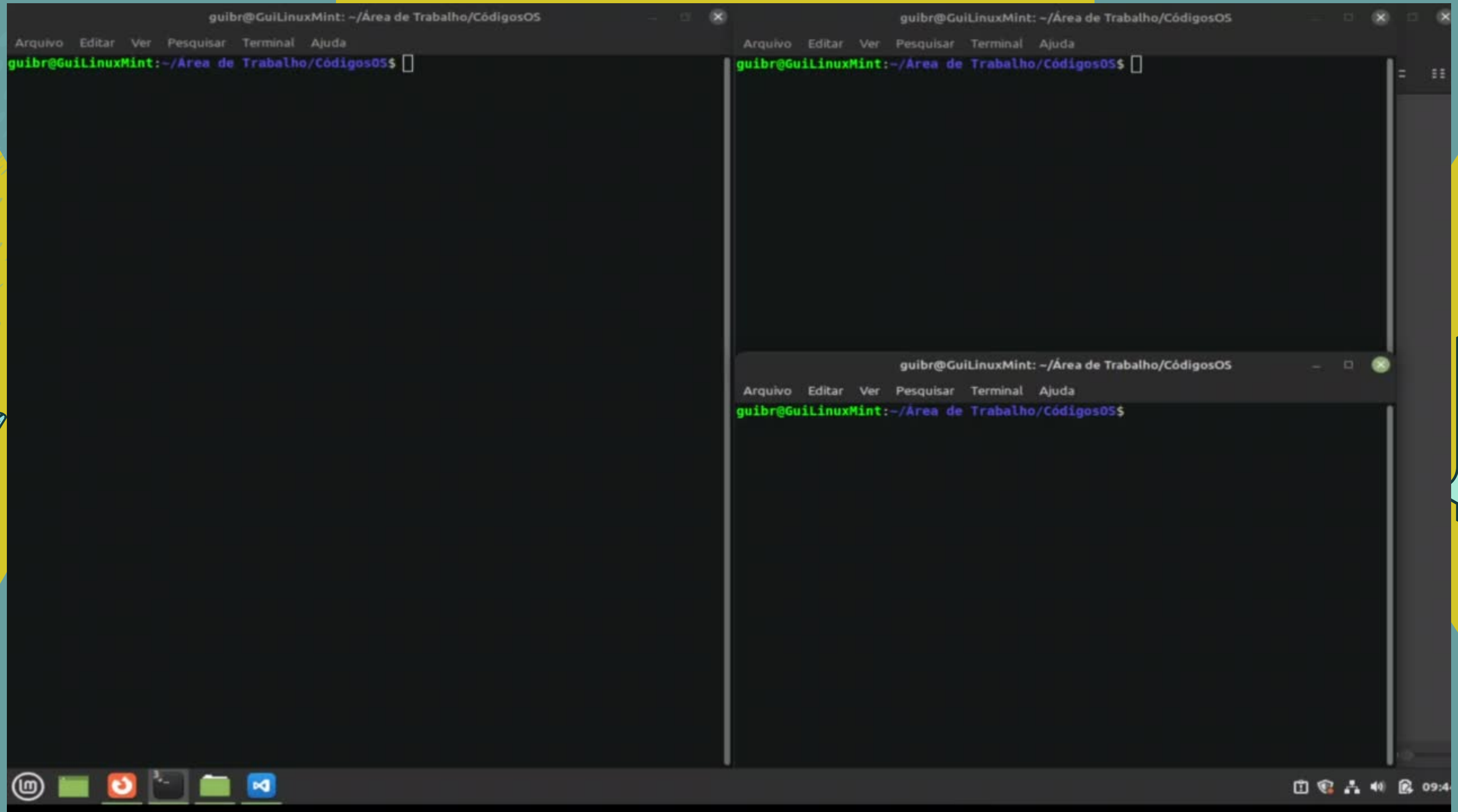
```
servidor.c - Visual Studio Code
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda

C cliente.c C servidor.c X
home > guibr > Área de Trabalho > CódigosOS > C servidor.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/socket.h>
6  #include <sys/types.h>
7  #include <netinet/in.h>
8  #include <arpa/inet.h>
9
10 #define PORT 4950
11
12 int main()
13 {
14     // Declaração das variáveis utilizadas no processo
15     int sockfd, ret;
16     struct sockaddr_in serverAddr;
17
18     int newSocket;
19     struct sockaddr_in newAddr;
20
21     socklen_t addr_size;
22
23     char buffer[1024];
24     pid_t childpid;
25
26     sockfd = socket(AF_INET, SOCK_STREAM, 0);
27     if(sockfd < 0){
28         printf("[-] Erro na conexão.\n");
29         exit(1);
30     }
31     printf("[+] Socket do cliente criado!\n");
32
33     // Definição e configuração de parâmetros de endereçamento
34     memset(&serverAddr, '\0', sizeof(serverAddr));
35     serverAddr.sin_family = AF_INET;
36     serverAddr.sin_port = htons(PORT);
37     serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
38
39     // Associa o socket com o endereço local
```

Ln 13, Col 55 Tamanho da Tecla Tab: 4 UTF-8 LF C

RODANDO

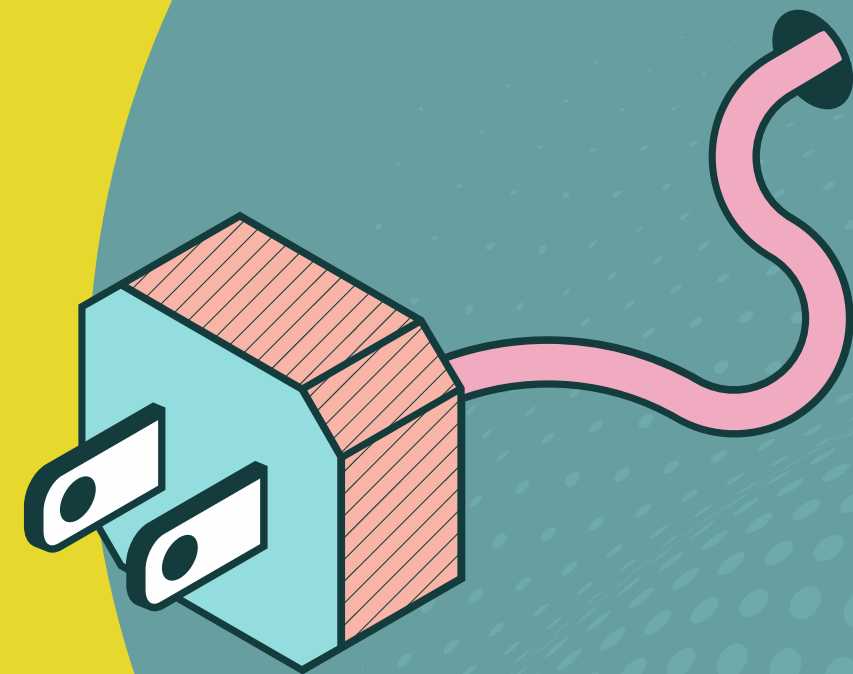
8



COMANDO

--LOG, --PORT E --HOST

Por meio do ARGV e ARGV, podemos criar um for para cada elemento que eles lerem e assim ver se algum desse elemento, ou desses elementos, pertence aos comandos e assim ele consegue responder o comando solicitado pelo Cliente.



COMPRESSÃO

A compressão de dados é o ato de reduzir o espaço ocupado por dados num determinado dispositivo. Essa operação é realizada através de diversos algoritmos de compressão, reduzindo a quantidade de Bytes para representar um dado, sendo esse dado uma imagem, um texto, ou um arquivo (ficheiro) qualquer.

- Pacote Zlib
- --Compress
- --Log





The logo consists of a central yellow circle containing the text 'Sistemas Operacionais UFRR'. This circle is surrounded by a thick light blue ring, which is further enclosed by two thin yellow concentric circles. Three small pink dots are positioned in the upper left area of the yellow circle.

**Sistemas
Operacionais
UFRR**

Obrigado!