

# **Relatório sobre a implementação da minha aplicação**

## **\*Bibliotecas utilizadas:**

-Stdio.h: Esta biblioteca contém várias funções de entrada e saída.

-Stdlib.h: Ela possui funções envolvendo alocação de memória, controle de processos e conversões.

-Unistd.h: Utilizada para obter acesso a arquivos e diretórios.

-String.h: Utilizada para manipulação de strings.

-Errno.h: Fornece macros para identificar e relatar erros de execução através de códigos de erro.

-Sys/types.h: Utilizada na manipulação de ficheiros

-Sys/socket: Biblioteca utilizada para manipulação de sockets

-Netinet/in.h: Define os tipos de dados para representar os endereços de socket

-Netdb.h: Internamente, o sistema usa um banco de dados para acompanhar o mapeamento entre nomes de host e números de host. Este banco de dados é geralmente o arquivo /etc/hosts fornecido por um servidor de nomes. As funções e outros símbolos para acessar este banco de dados são declarados em netdb.h.

## **Lado Servidor:**

### **\*Definindo Constantes:**

Inicialmente precisei definir qual o tamanho do buffer que utilizaria para a mensagem do cliente, por exagero da minha parte coloquei 8192, logo depois fui definir a porta a qual a comunicação aconteceria nesse caso a 9191.

### **\*Definindo Variáveis:**

Iniciei pela criação de duas estruturas já definidas pelas bibliotecas utilizadas `sockaddr_in` denominadas “cliente” e “servidor”, logo depois declarei as descrições do cliente e servidor denominadas “desc\_servidor” e “desc\_cliente”, seguindo mais a frente realizei a criação de uma string com o tamanho do buffer o qual armazenará a mensagem do cliente denominada “buffer”, logo mais declarei uma variável inteira chamada “yes” o qual ao longo do código atribui o número 1 para simbolizar que era um valor “true” em linguagem booleana por último mas não menos importante declarei outra variável cujo tipo “`socklen_t`” já vem das bibliotecas importadas chamada “cliente\_len” junto de uma variável inteira que será atribuída com o tamanho da mensagem, essa mesma intitulada “message\_len”.

### **\*Execução do código :**

A cada ponto do código existe um comentário explicando o que está acontecendo em cada momento, porém caso não deseje olhar diretamente o código venho através deste explicar melhor o funcionamento do código. Primeiramente imprime na tela uma mensagem de servidor iniciado logo depois parto para a criação de um socket no estilo IPv4, caso retorne -1 na criação significa que houve um problema com a mesma, então imprime uma mensagem de erro e finalizo a execução do programa. Caso criado com sucesso imprimo uma mensagem avisando que foi criado com sucesso, logo depois começo a definir as propriedades do socket, logo depois defino a variável “yes” como true ou seja = 1 e verifico as opções do socket caso retorne -1 da função “`setsockopt`” imprimo que aconteceu um erro e finalizo a execução do programa, feito isso eu tento ligar o socket com a porta que defini como constante caso não consiga ligar os dois faço o mesmo procedimento para todos os erros imprimo que aconteceu um erro e finalizo a execução do programa. Bom agora tento a escutar o cliente caso der erro já sabe, se tudo der certo imprimo que estou escutando na porta declarada como constante, atribuo o tamanho de um cliente para a variável “cliente\_len” e verifico se está aceitando a comunicação com o cliente, caso não esteja acontece o mesmo que com os outros erros, se deu certo imprime uma mensagem de boas vindas e envio uma mensagem para o cliente, para que o mesmo saiba que está conectado, agora inicia-se um laço que fica recebendo as mensagens do cliente até que o mesmo diga “bye”, sempre limpando o buffer a cada iteração. Assim que recebido a mensagem “bye” termino a conexão com o cliente e finalizo o socket que foi criado localmente

# **Lado Cliente:**

## **\*Definindo Constantes:**

Inicialmente precisei definir o tamanho do buffer que foi declarado no servidor para a mensagem denominada "LEN", por exagero da minha parte coloquei 8192, logo depois fui definir a porta a qual foi definida no lado do servidor para a comunicação ou seja 9191.

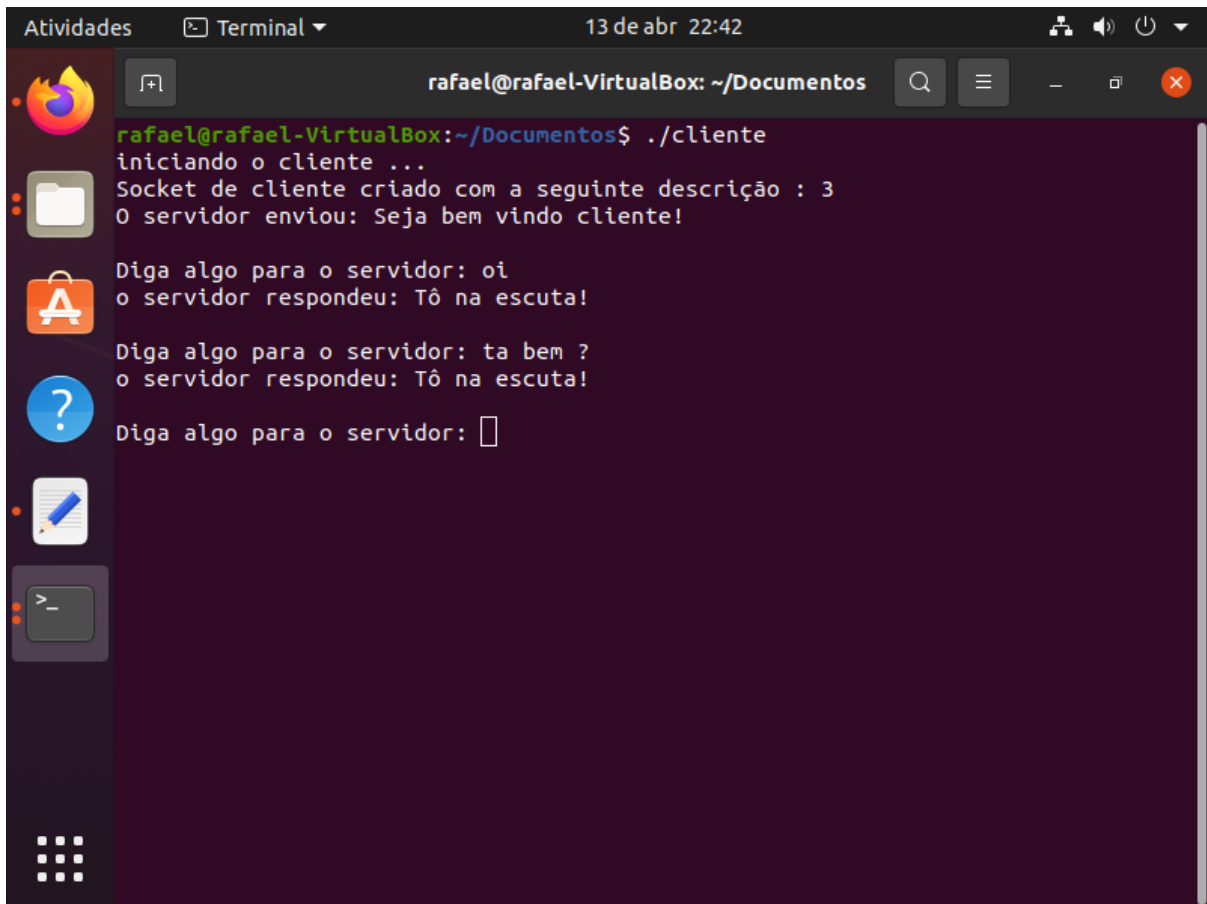
## **\*Definindo Variáveis:**

Definido inicialmente a "sockaddr\_in" chamada de servidor, uma variável inteira para a descrição do socket chamada "desc\_sock", uma denominada "len com o tamanho da variável antes declarada chamada "servidor", depois declaro uma variável inteira chamada "slen" que irá armazenar o tamanho da mensagem recebida pelo servidor, logo mais venho a declarar uma string de buffer de entrada e uma de buffer de saída com o tamanho da constante definida anteriormente "LEN".

## **\*Execução do código :**

Começo o código imprimindo que estou iniciando o cliente, logo depois parto para a criação do socket de cliente, caso de qualquer erro faço do mesmo jeito que no servidor : imprimo o que deu errado e finalizo a execução do programa, coloco na tela que o socket de cliente foi criado com a descrição "3", tento definir as propriedades da conexão, faço a tentativa de conexão com o servidor, o cliente recebe a mensagem de bem vindo do servidor, mantém a conexão até a palavra "bye" ser transmitida, dentro deste laço primeiro eu reseto os buffers de entrada e saída, realizo a leitura da mensagem a ser enviada para o servidor, logo depois escuto o que o servidor tem a responder, no final do laço e venho a verificar se a mensagem recebida foi "bye" caso sim, saio do laço, para finalizar eu fecho a conexão e coloco na tela a informação que a conexão foi encerrada

## Exemplo de conversa entre cliente e servidor:



The screenshot shows a terminal window titled "rafael@rafael-VirtualBox: ~/Documentos" with a dark purple background. The terminal output shows the execution of a client program and its interaction with a server. The user enters the command `./cliente`, which outputs "iniciando o cliente ...". The server then sends a message: "Socket de cliente criado com a seguinte descrição : 3" and "O servidor enviou: Seja bem vindo cliente!". The user then enters "oi", and the server responds "o servidor respondeu: Tô na escuta!". The user enters "ta bem ?", and the server responds "o servidor respondeu: Tô na escuta!". Finally, the user enters a blank line, and the prompt "Diga algo para o servidor: " is shown with a cursor.

```
rafael@rafael-VirtualBox: ~/Documentos$ ./cliente
iniciando o cliente ...
Socket de cliente criado com a seguinte descrição : 3
O servidor enviou: Seja bem vindo cliente!

Diga algo para o servidor: oi
o servidor respondeu: Tô na escuta!

Diga algo para o servidor: ta bem ?
o servidor respondeu: Tô na escuta!

Diga algo para o servidor: 
```

Atividades Terminal 13 de abr 22:41

rafael@rafael-VirtualBox: ~/Documentos

```
rafael@rafael-VirtualBox:~/Documentos$ ./servidor
Servidor Iniciado
O socket do servidor foi criado com a seguinte descrição: 3
Ouvindo a porta: 9191
O cliente esta conectado.
No aguardo de alguma mensagem do cliente ...
O nosso querido cliente disse: oi
O nosso querido cliente disse: ta bem ?
█
```