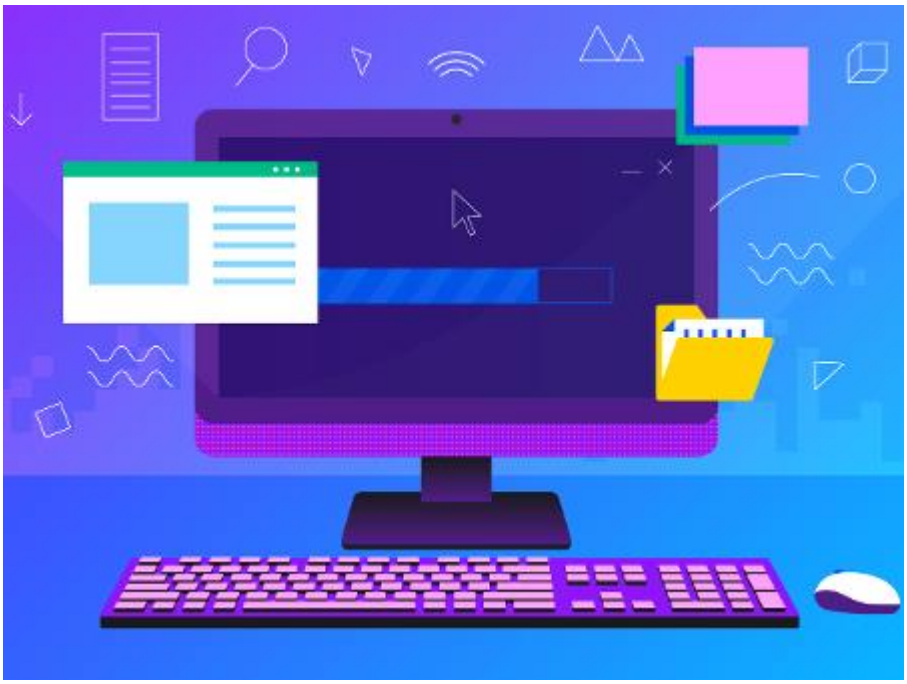
Universidade de Évora

Curso de Engenharia Informática

Redes de Computadores 2020/2021

Trabalho prático



Trabalho realizado por:

Dinis Matos nº42738

José Lopes nº37861

Introdução

Este trabalho teve como objetivo o desenvolvimento de um programa em C que represente um "client" que vai comunicar com um "server", também um programa em C. O servidor é suposto receber vários comandos do cliente, e executar as tarefas pedidas. O Admin tem como “default” o username: “ADMIN” e password: “admin”.

Estrutura do trabalho

server.c

struct client

Esta struct vai guardar todas as informações relevantes de um cliente, quando está ligado ao servidor, através de várias variáveis.

struct replyPacket

Esta struct vai guardar um leque de strings que formam a resposta do servidor para o cliente, a packet de resposta.

struct replyPacket initializereplyPacket

Este método vai inicializar todas as strings na reply packet com "".

void printPacket

Este método vai imprimir todo o conteúdo da reply packet.

void initializeClientDB

Este método vai inicializar a primeira posição do array "ClientDB" (array com todas as structs que reprentam todos os utilizadores ligados ao servidor) com o utilizador "ADMIN" e a sua respetiva password, e vai popular todas as posições restantes com valores default ou com informação lida de um ficheiro local.

void saveClientDB

Este método vai guardar todas as informações no array "ClientDB" num ficheiro local, “database.txt”.

void printClientDB

Este método vai imprimir todas as structs da "ClientDB".

void setClient

Este método vai receber os dados sobre o cliente e vai popular a sua respetiva struct no array "ClientDB".

void startupClient

Este método vai fazer o setup inicial de um cliente.

int findClientStructByNick

Este método vai retornar o índice de um cliente no array "ClientDB", procurando pelo seu nickname.

int findClientStructBySocket

Este método vai retornar o índice de um cliente no array "ClientDB", procurando pela sua socket.

int findEmptySpace

Este método vai encontrar um espaço não ocupado no array de clientes.

void splitStrings

Este método vai dividir o input do cliente pelos espaços.

bool isValidNick

Este método vai ver se o nickname introduzido através do comando NICK é valido.

bool isOnline

Este método vai verificar se o cliente está "online" (através dos parâmetros definidos).

bool isChannelValid

Este método vai verificar se o canal para qual o cliente está a aceder é valido.

void allChannels

Este método vai colocar os nomes de todos os canais válidos numa string "aux".

void clientStatus

Este método vai retornar o estatuto que o cliente tem no servidor.

void allWhos

Este método vai retornar os nomes de todos os outros clientes que estão no canal atual do cliente.

Métodos correspondentes aos comandos do cliente

De seguida vão ser definidos todos os métodos que foram pedidos no enunciado, como o NICK, MSSG, etc.

struct replyPacket methodSelector

Método que vai tratar de enviar as mensagens de resposta para os clientes, após algum destes ter utilizado um dos comandos. Ou manda uma mensagem de sucesso, ou manda a respetiva exceção.

void bridge

Finalmente, existe o método "bridge" que é o método que vai tratar da comunicação entre cliente e servidor.

int main

Função principal do servidor que é responsável por todo o processo de comunicação entre o cliente e o servidor.

cliente.c

int main

Função do cliente que é responsável por todo o envio de comandos por parte do utilizador e que o mesmo aceita resposta do servidor. O cliente pode parar a comunição ao colocar “exit” ou receber novas informções ao colocar “update”.

Para compilar o cliente utiliza-se: gcc client.c -o client. Para compilar o server utiliza-se: gcc server.c -o server. Depois basta executar cada um em terminais diferentes, ou seja, ./client e ./server.