

Trabalho Prático no âmbito da cadeira Redes de Comunicação

**Meta Final**

**Sistema de turmas com uso dos protocolos TCP/UDP**

**Realizado por:**

**Miguel Curto Castela - uc2022212972**

**Francisco Lapa Silva - uc2022213583**

Índice

# 

Conteúdo

[Funcionalidades do sistema 3](#_Toc167040919)

[class\_server.c 3](#_Toc167040920)

[main 3](#_Toc167040921)

[funções 3](#_Toc167040922)

[file\_manager.c 3](#_Toc167040923)

[funções 3](#_Toc167040924)

[class\_struct.c 4](#_Toc167040925)

[funções 4](#_Toc167040926)

[Commands\_server.c 4](#_Toc167040927)

[funções 4](#_Toc167040928)

[class\_client.c 4](#_Toc167040929)

[main 4](#_Toc167040930)

[funções 5](#_Toc167040931)

[Class\_admin.c 5](#_Toc167040932)

[main 5](#_Toc167040933)

[Comandos de utilizador 5](#_Toc167040934)

[Configuração de rede com o GNS3 6](#_Toc167040935)

[Comandos de configuração usados para os dispositivos 6](#_Toc167040936)

Notas:

* Como alternativa ao Netcat, foi feita uma aplicação cliente UDP.
* Um makefile foi incluído no servidor e em cada PC para facilitar a compilação.

# Funcionalidades do sistema

## class\_server.c

### main

* É feito o controlo dos argumentos de entrada, ou seja, os portos das turmas e o ficheiro de configurações com as informações dos users.
* São validados os portos e os argumentos iniciais.
* O SIGINT é redirecionado para controlar o encerramento (fechar o main e as conexões TCP e UDP, desligar o sistema, terminar os processos filho e libertar recursos partilhados). As funções usadas neste efeito são as system\_shutdown, close\_main e close\_tcp
* um semáforo é usado para controlar acesso a cada memória partilhada e dois ao ficheiro de configuração.
* É criada a memória partilhada (com create\_shared\_memory) para armazenar informações de turmas e PIDs dos processos filhos.
* Inicializa semáforos para sincronizar o acesso aos recursos partilhados e controla as conexões UDP e TCP.
* São criadas as threads para as conexões TCP e UDP. (com o handle\_tcp e handle\_udp)

### funções

O servidor cria e gerencia threads separadas para lidar com conexões TCP (os usuários) e UDP (os administradores), aceita novas conexões TCP e cria um novo processo para cada cliente. Cada um desses processos filhos controla a conexão com os clientes(através de process\_client\_tcp) e interpreta ,responde e formata as requests dos clientes, (através do handle\_requests\_tcp e handle\_usercursor) incluindo o login, logout, listagem de turmas, inscrição em turmas e envio de mensagens. A thread UDP controla a connecção UDP (através de process\_admin\_udp) e interpreta, responde e formata os requests dos administradores (através de handle\_request\_udp e handle\_usercursor),incluindo o login, logout, listagem de comandos, adicionar remover e listar utilizadores e desligar o servidor.

## file\_manager.c

* Controla e valida as informações dos utilizadores guardados no ficheiros de configuração.
* São utilizandos semáforos para a sincronizaçãode acesso ao ficheiro.

### funções

As principais funções incluem file\_checkintegrity (chamada no main , no início do programa, no processo de validação dos input), que verifica a integridade do ficheiro de configuração. file\_finduser, que procura um utilizador verificando o nome de utilizador e a palavra-passe específica no ficheiro, e devolvendo o tipo (utilizado para login). file\_adduser, que adiciona um novo utilizador ao ficheiro, validando previamente se este já existe e se o tipo é válido, e file\_removeuser, que remove um utilizador do ficheiro.

Adicionalmente, a função file\_listusers lista, de forma formatada, todos os utilizadores presentes no ficheiro.

## class\_struct.c

* Uma struct de turma tem os campos **name, size, subscribed, subscribed\_names, mutilcast\_addr, udp\_socket**

### funções

Uma struct turma é criada e inicializada com a função create\_classstruct. Esta valida também se a estrutura já está ocupada e se o tamanho da turma é válido. A função destroy\_classtruct verifica que a turma não está vazia e por fim dá reset aos campos da struct. A função addsub\_classtruct tenta adicionar um aluno a uma turma , verificando primeiro se há espaço e se ele já está inscrito. A função sendmsg\_classtruct escreve para o endereço multicast da turma (**multicast\_addr**) a mensagem dada. Cada chamada desta função abre e configura um novo socket com o dado endereço, encerrando após a mensagem ser enviada.

## Commands\_server.c

* Todas as funções descritas neste ficheiro são chamadas pelas funções handle\_request\_tcp e handle\_request\_udp presentes no ficheiro Class\_server.c.
* Todas estas funções recebem os inputs necessários para funcionarem e um ponteiro **\*response** para onde vão escrever (num **buffer**) a informação a enviar ao utilizador

### funções

|  |  |  |
| --- | --- | --- |
| \*\*login | \*\*add\_user | \**list\_users* |
| Tenta autenticar um utilizador com o nome e password dados, devolvendo o tipo, se as credenciais estiverem corretas | Adiciona um novo utilizador ao ficheiro de configuração, com as credenciais dadas. | Lista todos os utilizadores e os seus tipos do ficheiro de configuração |
| \*\*list\_classes | \*\*list\_subscribe | \*\*subscribe\_class |
| Lista todas as aulas existentes. | Lista todas as aulas em que o utilizador está inscrito. | Inscreve o utilizador na turma com o nome dado. |
| \*\*create\_class | \*\*send\_message | \**list\_cmds\_udp* |
| Cria uma turma com o nome e tamanho dados. | Envia uma mensagem para o endereço multicast da turma dada. | Lista todos os comandos UDP disponíveis. |
| \*\*logout | \*\*del\_user | \*list\_commands\_tcp |
| Dá reset à struct User, o que retira as permissões de User ao cliente | Apaga um utilizador, com o nome dado, do ficheiro de configuração. | Lista todos os comandos TCP disponíveis. |

## class\_client.c

### main

* Faz o controlo de elementos de entrada (IP do servidor e porto TCP a que está conectado)
* verifica que o ip é válido (a partir do gethostbyname()) e que o port é válido (entre 1024-65535).
* Abre o **socket** e tenta conectar ao endereço e porta especificados no servidor.
* Após entrar, fica num loop para lidar com a comunicação com o servidor, para registar a entrada do aluno e ler mensagens deste (através dos **buffers** para entrada e saída de mensagens).
* Sempre que é recebida uma mensagem TCP do servidor, o cliente verifica se é uma mensagem especial antes de escrever para o ecrã e esperar resposta do utilizador (mensagens especiais :"-+!SERVER-CL0SING!+-"-> quebra o loop e fecha o cliente, o servidor fecha "-+!L0G0UT!+-" -> o servidor deu ordem de saída a todos os grupos multicast em que este utilizador estava previamente, espera nova mensagem do servidor antes de continuar "-+!MULT1C4ST!+- XXX.XXX.XXX.XXX" -> o servidor ordena que este cliente se junte ao grupo multicast do endereço dado e espera nova mensagem do servidor antes de continuar). Isto é feito para o servidor controlar algumas funções do cliente.

### funções

Garante que o cliente envia uma mensagem de saída ao servidor e fecha o **socket** antes de sair através da handle\_sigint. A função \*muliticast\_listener ouve e espera ,em paralelo do main, por threads, por mensagens multicast de um dado IP, escrevendo-as no terminal.

## Class\_admin.c

### main

* Faz o controlo de elementos de entrada (IP do servidor e porto UDP a que está conectado)
* Cria um socket UDP e inicializa a estrutura **server\_address** (com os campos **tipo de conexão,** endereço **IP** e **porta do servidor**
* Redireciona o SIGINT para assegurar que o socket é fechado corretamente
* Envia uma mensagem de "HELLO" ao servidor para iniciar a comunicação, depois entra no loop de esperar respostas, escreve-las na consola e esperar e enviar novos pedidos do utilizador

# Comandos de utilizador

|  |  |  |  |
| --- | --- | --- | --- |
| **ADMIN** | | **COMANDOS EXCLUSIVOS A ALUNOS** | |
| **ADD USER:** | Adiciona um novo utilizador (**type** pode ser aluno, professor ou admin) | **SUBSCRIBE\_CLASS** | Inscreve o usuário em uma classe. |
| **LIST:** | Lista todos os utilizadores do sistema | **CREATE\_CLASS** | Permite que um professor crie uma classe. |
| **DEL <username> :** | Apaga um utilizador | **LIST\_SUBSCRIBED** | Lista todas as classes às quais o usuário está inscrito. |
|  | | **COMANDOS EXCLUSIVOS A PROFESSORES** | |
| **QUIT\_SERVER:** | Fecha a sessão de admin e ordena o encerramento do servidor | **SEND** | Permite que um professor envie uma mensagem para uma classe. |
| **LIST\_CLASSES** | Lista todas as classes disponíveis. |
|  | | **COMANDOS DE PROFESSORES E ALUNOS** | |
| **LOGOUT:** | Dá logout ao admin atual. | Faz logout do usuário | |
| **LOGIN <user\_name> <password>:** | Autentica o administrador | Autentica o utilizador e ordena-o a juntar-se aos grupos multicast das turmas em que ele já está inscrito | |
| **HELP :** | Lista os comandos disponíveis para o admin | Lista os comandos disponíveis para os professores ou alunos | |

# Configuração de rede com o GNS3

Uma imagem com texto, diagrama, captura de ecrã, file

Descrição gerada automaticamente

## Uma imagem com texto, captura de ecrã, Tipo de letra Descrição gerada automaticamenteComandos de configuração usados para os dispositivos

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

**Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente**

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, cartão de visita, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente