

Trabalho Prático no âmbito da cadeira Redes de Comunicação

**Meta Final**

**Sistema de turmas com uso dos protocolos TCP/UDP e multicast.**

**Realizado por:**

**Miguel Curto Castela - uc2022212972**

**Francisco Lapa Silva - uc2022213583**

Índice

# 

Conteúdo

[1. Funcionalidades do sistema 3](#_Toc167044509)

[1.1 class\_server.c 3](#_Toc167044510)

[main 3](#_Toc167044511)

[funções 3](#_Toc167044512)

[1.2 file\_manager.c 3](#_Toc167044513)

[funções 3](#_Toc167044514)

[1.3 class\_struct.c 3](#_Toc167044515)

[funções 3](#_Toc167044516)

[1.4 Commands\_server.c 4](#_Toc167044517)

[funções 4](#_Toc167044518)

[1.5 class\_client.c 4](#_Toc167044519)

[main 4](#_Toc167044520)

[funções 4](#_Toc167044521)

[1.6 Class\_admin.c 4](#_Toc167044522)

[main 4](#_Toc167044523)

[2. Comandos de utilizador 5](#_Toc167044524)

[3. Configuração de rede com o GNS3 5](#_Toc167044525)

[3.1 Comandos de configuração usados para os dispositivos 6](#_Toc167044526)

Notas:

* Como alternativa ao Netcat, foi feita uma aplicação cliente UDP (class\_admin).
* Um makefile foi incluído no servidor e em cada PC para facilitar a compilação (make server; make client; make admin).

# Funcionalidades do sistema

## class\_server.c

### 

### main

* É feito o controlo dos argumentos de entrada, ou seja, as *ports* das turmas e o ficheiro de configurações com as informações dos *users*.
* São validados as *ports* 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 o 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 gere *threads* separadas para lidar com conexões TCP (os utilizadores) e UDP (os administradores), aceita novas conexões TCP e cria um processo para cada cliente. Cada um desses processos filho controlam a ligação 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 ligação 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, listar utilizadores e desligar o servidor.

## file\_manager.c

* Controla e valida as informações dos utilizadores guardados nos ficheiros de configuração.
* São utilizados semáforos para a sincronização de acesso ao ficheiro.

### funções

As principais funções incluem file\_checkintegrity (chamada no *main* localizado no início do programa, no processo de validação dos *inputs*), 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 reinicia os campos da *struct*. A função addsub\_classtruct tenta adicionar um aluno a uma turma, verificando primeiro se há espaço e se este 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 este 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 | create\_class | list\_users |
| Tenta autenticar um utilizador com o nome e password dados, devolvendo o tipo, se as credenciais estiverem corretas | Cria uma turma com o nome e tamanho dados. Atribui-lhe o próximo endereço multicast disponivel | Lista todos os utilizadores e os seus tipos a partir do ficheiro de configuração. |
| list\_classes | list\_subscribe | add\_user |
| Lista todas as aulas existentes. | Lista todas as aulas em que o utilizador está inscrito. | Adiciona um novo utilizador ao ficheiro de configuração, com as credenciais dadas. |
| subscribe\_class | send\_message | *list\_cmds\_udp* |
| Inscreve o utilizador na turma com o nome dado. | 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 port TCP a que está ligado).
* Verifica que o IP é válido (a partir do gethostbyname()) e que a *port* é válido (entre 1024-65535).
* Abre o **socket** e tenta ligar ao endereço e port 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 também fecha "-+!L0G0UT!+-" -> o servidor deu ordem de saída a todos os grupos multicast em que este utilizador estava previamente e espera por 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 que o servidor possa 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*, a partir de *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 port UDP a que está ligado).
* Cria um *socket* UDP e inicializa a estrutura **server\_address** (com os campos **tipo de ligação,** endereço **IP** e **port 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 e enviar pedidos do utilizador e receber e escrever resposta do servidor.

# Comandos de utilizador

|  |  |  |  |
| --- | --- | --- | --- |
| **ADMIN** | | **COMANDOS EXCLUSIVOS A ALUNOS** | |
| **ADD USER <user\_name> <password> <type>:** | Adiciona um novo utilizador (**type** pode ser aluno, professor ou admin) | **SUBSCRIBE\_CLASS<class\_name>:** | Inscreve o utilizador numa turma. |
| **LIST:** | Lista todos os utilizadores do sistema | **CREATE\_CLASS <class\_name> <size>:** | Permite que um professor crie uma turma. |
| **DEL <username>:** | Apaga um utilizador | **LIST\_SUBSCRIBED** | Lista todas as classes às quais o utilizador está inscrito. |
|  | | **COMANDOS EXCLUSIVOS A PROFESSORES** | |
| **QUIT\_SERVER:** | Fecha a sessão de admin e ordena o encerramento do servidor | **SEND <class\_name> <message>:** | Permite que um professor envie uma mensagem para uma turma. |
| **LIST\_CLASSES** | Lista todas as turmas disponíveis. |
|  | | **COMANDOS DE PROFESSORES E ALUNOS** | |
| **LOGOUT:** | Dá logout ao admin atual. | Faz logout do utilizador. | |
| **LOGIN <user\_name> <password>:** | Autêntica o administrador | Autêntica 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 automaticamenteUma imagem com texto, captura de ecrã, cartão de visita, Tipo de letra

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

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

Descrição gerada automaticamente