Relatório - Projeto - Redes de comunicação 2023/24
João Vaz e André Teixeira

Inicialmente quando iniciamos o nosso server(class_server <porta_tcp <porta_udp> <ficheiro.txt>, é feita a inicialização de uma shared memory que nos irá permitir gerir e armazenar as informações de todos os utilizadores e turmas. Após isso é feita a leitura do ficheiro que contém já os utilizadores existentes e que são carregados para a shared memory anteriormente mencionada. Depois, esperamos tanto por ligações tcp como udp, tcp por parte dos professores e alunos e udp para os administradores (que podem ligar-se ao server através do netcat). Temos duas funções: handle_connection_tcp e handle_connection_udp , responsáveis por receber o input dos utilizadores e chamar a função correspondente que irá processar esse input (verifica_comando_cliente e verifica_comando_admin) . Dependendo do comando recebido essas funções irão reencaminhar o comando para outras funções, essas responsáveis por apresentar informação ou fazer as alterações propriamente ditas dependendo do comando recebido.

Os clientes (professores e alunos) devem após executar o executável class_client <ip> <porta> devem fazer login através do comando LOGIN <username> <password>.

A função verify_login_tcp irá então verificar se o username introduzido existe e se a password está correta, para além disso irá retornar um inteiro, 2 se for um professor e 1 se for um aluno e que servirá para atualizar uma flag que permitirá distinguir os comandos que o professor e os alunos têm acesso.

Após o login bem sucedido os clientes têm acesso a vários comandos:

LIST_CLASSES

Tanto para os alunos como para os professores. Apresenta as turmas existentes.

LIST_SUBSCRIBED

Tanto para os alunos como para os professores. Apresenta as turmas em que o aluno ou o professor está inscrito.

João Vaz - 2022231087
André Teixeira - 2022231134

SAIR

Tanto para os alunos como para os professores. Sai. Faz logout.

CREATE_CLASS <nome> <size>

Exclusivo para os professores. Cria uma turma com o nome passado e com o tamanho máximo de <size>

SUBSCRIBE_CLASS <nome>

Exclusivo para os alunos. Inscreve o aluno na turma.

SEND <nome> <mensagem>

Exclusivo para os professores. Envia o conteúdo de mensagem para todos os alunos inscritos na turma passada por parâmetro.

COMANDOS EXCLUSIVOS DO ADMINISTRADOR

ADD_USER <username> <password> <cargo>

Permite adicionar um novo utilizador. Atualiza a shared memory e o ficheiro .txt.

DEL <username>

Permite remover um utilizador. Atualiza a shared memory e o ficheiro .txt.

LIST

Lista todos os utilizadores existentes.

QUIT_SERVER

Fecha o server.

Composição da nossa shared memory

```c
typedef struct {
    char username[MAX_USER_INFO_LENGTH];
    char password[MAX_USER_INFO_LENGTH];
    char role[MAX_USER_INFO_LENGTH];
} UserInfo;

typedef struct {
    char nome[BUFFER_SIZE];
}Aluno;

typedef struct {
    char name[MAX_USER_INFO_LENGTH];
    int n_alunos;
    int size;
    char multicast_address[16];
    Aluno alunos[100];
    char prof[BUFFER_SIZE];
} ClassInfo;

typedef struct{
    UserInfo users[MAX_USERS];
    ClassInfo classes[MAX_USERS];
    int n_users;
    int n_classes;
}sh_mem;

sh_mem *shared_mem;
```

A shared memory tem um array com os utilizadores todos , outro array com as turmas todas e duas variáveis do tipo inteiro que armazena o número de utilizadores e número de turmas existentes (bastante útil para percorrer os arrays).

João Vaz - 2022231087
André Teixeira - 2022231134

Funcionamento do multicast

Uma das funcionalidades mais importantes deste sistema é a capacidade de um professor poder enviar uma mensagem para todos os alunos nela inscritos através do comando SEND, que envia a mensagem para o endereço multicast já associado à turma. Por outro lado, quando um aluno usa o comando SUBSCRIBE_CLASS, ou seja inscreve-se numa turma, é iniciada uma thread que fica à espera de receber comunicações através do endereço multicast passado por parâmetro para a função da thread.