Redes de Computadores - Trabalho Prático

Autor:
Vasco Barnabé - l 42819

15 de Fevereiro de 2021

1 Introdução

Este trabalho tem como objetivo o desenvolvimento de um cliente e de um servidor de acordo com determinado protocolo. É pedido que o servidor consiga gerir vários clientes em simultâneo usando select (sem utilizar fork ou threads) e será responsável pela gestão:

- de receção e re-envio das mensagens;
- dos canais;
- dos dados dos utilizadores, podendo os clientes inserir, ler mensagens, e alterar os dados na sua conta/perfil.

O servidor tem 3 canais, um canal "default", um canal "cn"(computer networks) e um canal "oss"(open source software).

Relativamente aos clientes, estes podem funcionar em dois modos: o modo registado e não registado, em que qualquer um deles é obrigatório ter um nickname/nome definido (que funcionará como um username visível e que é único no servidor). Se o cliente não estiver registado, apenas tem permissão para funcionar no canal default, enquanto que se estiver registado poderá iniciar no seu canal preferido, que pode ser alterado a qualquer instante. Se o cliente estiver registado, este poderá ainda gerir canais e gerir outros utilizadores, adquirindo assim o estatuto de operador.

2 Implementação

A implementação deste trabalho foi dividida em 4 seções, consistindo assim em:

- **Servidor**(ficheiro "server.c") - A implementação do servidor é uma das tarefas mais importantes neste projeto, tendo sido aqui implementado o seu mecanismo e todos os comandos disponíveis no chat para o utilizador, com algumas particularidades.

  Comandos implementados:

1. **NICK** - Este comando é um comando de utilizador e pode ser usado por todos os utilizadores, sejam estes registados ou não. Este comando atribui ou muda o nome/nickname do utilizador, com algumas condições, sendo apresentadas mensagens "erro"sempre que o utilizador não insere o nickname após o comando NICK ou sempre que o nickname inserido já exista. A função de surgir uma mensagem de erro sempre que o nickname tenha 10 ou mais caracteres não está funcional. Se o nome for aceite, o utilizador é notificado de que realizou a tarefa com sucesso.
2. **MSSG** - Este comando é também um comando de utilizador, estando, tal como o comando anterior, disponível pra todos os utilizadores. Sempre que o utilizador pretender enviar uma mensagem (para o canal ativo), recorre a este comando digitando em conjunto a mensagem que pretende enviar. O programa verifica se utilizador digitou alguma mensagem ou se esta é demasiado longa (exceder os 512 bytes). Em caso de o utilizador não ter digitado a mensagem ou esta exceder o tamanho máximo definido, é notificado e tem de escrever nova mensagem.
3. **PASS** - Este comando apenas pode ser utilizado por utilizadores registados (podem ser operadores ou não), servindo para o utilizador se autenticar, sendo sempre notificado se foi autenticado com sucesso ou se ocorreu algum tipo de erro, como o seu nickname não estar definido ou a password inserida estar incorreta.
4. **JOIN** - Comando apenas disponível para utilizadores registados que pretendam alterar o canal ativo. Sempre que este comando é executado com sucesso, o utilizador é notificado de tal situação e o servidor notifica todos os utilizadores no novo canal de que o novo utilizador entrou no canal e notifica também os utilizadores do canal anterior que o utilizador deixou o canal. Podem também surgir mensagens de erro se o utilizador pretender mudar-se para um canal que não existe, ou se o utilizador não estiver registado.
5. **LIST** - Este comando apresenta ao utilizador todos os canais existentes (3, nesta implementação). Este é também um comando apenas disponível para utilizadores registados.
6. **WHOS** - Comando que quando invocado pelo utilizador disponibiliza a este informação relativa a todos os utilizadores conectados aquele canal, informações como os nicknames dos utilizadores, se estão ou não registados e se são ou não operadores.
7. **KICK** - Comando apenas disponível para utilizadores que são operadores, que quando invocado retira da lista dos utilizadores registados o utilizador mencionado.
8. **REGS** - Este comando regista um novo utilizador, com nickname e password, adicionando-o à lista dos utilizadores registados.
9. **OPER** - Este comando promove o utilizador em questão a operador (é condição prévia que o utilizador esteja registado).
10. **QUIT** - Comando invocado pelo utilizador que é operador e pretende deixar de o ser.

Nota: Os comandos **KICK**, **REGS**, **OPER**, **QUIT** não se encontram funcionais devido a um problema na leitura do ficheiro onde se encontram os utilizadores que são operadores. Não foi possível encontrar o final de uma linha no ficheiro, e por isso tornou-se impossível a leitura dos utilizadores operadores, e consequentemente, a verificação de se o utilizador era operador ou não. Por isto, os comandos já referidos e os erros a eles associados foram implementados, mas não se encontram funcionais.

- **Cliente**(ficheiro "cliente.c") - Implementação do utilizador. Aqui foi implementada a **struct** cliente composta por variáveis onde serão guardadas informações do utilizador tais como o seu nickname, a password, se é operador e o canal onde está. Foram também implementados alguns métodos auxiliares de alguns comandos das mensagens, tais como:
  - **bool UserNameonline(char array)** - método utilizado no comando NICK para auxiliar no processo de verificação de se um utilizador com aquele nickname já é existente, já está online;
  - **bool UserNameregistado(char array)** - método que verifica se um utilizador está registado ou não, e por isso é utilizado em comandos exclusivos para utilizadores registados, como **JOIN**, **LIST** e **WHOS**.
  - **bool verificalogin(char array0, char array1)** - método auxiliar no comando **PASS**, verifica se o nickname e a password inseridos pelo utilizador existem e se estão presentes no ficheiro com os utilizadores registados.

  Nota: foram criados ainda outros métodos auxiliares que acabaram por não ser utilizados nesta implementação, tal como foram criados métodos que iriam ser auxiliares nos comandos exclusivos para operadores, mas devido a um erro não solucionado (acima referido), não estão funcionais.

- **Correção de Erros**(ficheiro "t1.c") - Uma vez que é necessário considerar que a comunicação não é fiável e existe uma pequena probabilidade de ocorrerem erros no envio que não são detetados nem corrigidos, foi necessário implementar um sistema de correção de erros usando código de Hamming. É também aqui que estão implementados os métodos responsáveis por encriptar e desencriptar mensagens transmitidas.

- **Chat**(ficheiro "exp1.c") - É aqui que se dá a trocas de mensagens por parte do utilizador, sendo estas, encriptadas, enviadas para os destinos pretendidos e recebidas para o utilizador destinatário.

3 Teste do Software

Para a compilação no gcc e teste do software, são necessárias 3 janelas do terminal em simultâneo, sendo que numa delas é compilado e executado o servidor, com os comandos **gcc -o server server.c -lm** e **./server**, respetivamente; e as duas restantes correspondem a utilizadores. Em cada janela de utilizador é necessário compilar e executar com os seguintes comandos: **gcc -o exp1 exp1.c -lm** e **./exp1**. Será então apresentada a mensagem "CHAT", indicando ao utilizador que entrou no chat e pode começar a comunicar, utilizando os comandos disponíveis.

Para finalizar o teste de software, utiliza-se o comando CTRL+C na janela do servidor e ambas as janelas de utilizadores cessam a sua funcionalidade.

4 Conclusão

Chegada a fase de conclusão deste relatório, importa referir que, de um modo geral, os principais objetivos deste trabalho foram atingidos.

Foi desenvolvido um servidor gerenciador de vários clientes em simultâneo, com várias funcionalidades, descritas acima na introdução deste relatório.

Aquando a sua implementação, que foi dividida em 4 seções, surgiram algumas dificuldades, algumas delas que não colocando em risco a funcionalidade principal do projeto, não foram possíveis de ser solucionadas, limitando por isso o seu funcionamento em pleno.

Como por exemplo, não ter sido possível executar uma leitura correta num ficheiro, o que impossibilitou a funcionalidade de alguns comandos de operador pretendidos para este projeto.

No entanto, outras dificuldades surgiram ao longo do desenvolvimento deste trabalho, mas que foram superadas, nomeadamente a troca de mensagens entre utilizadores num canal.