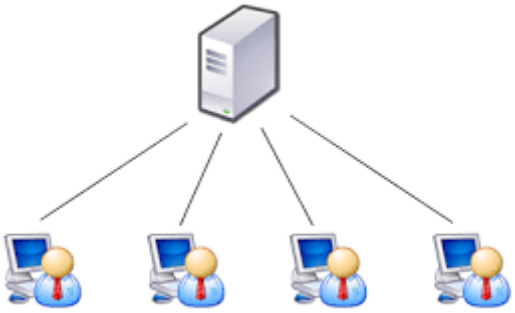
UNIVERSIDADE DE ÉVORA

2020/2021 – 5º Semestre

Relatório – Trabalho prático

REDES DE COMPUTADORES

INTERFACE DE COMUNICAÇÃO ENTRE CLIENTES E SERVIDOR



Discentes:

Pedro Patinho

Luís Rato

Trabalho realizado por:

Diogo Castanho, 42496

Pedro Grilo, 43012

1.Objetivo do trabalho prático

Foi-nos proposta a realização de um trabalho que consiste no desenvolvimento de uma interface cliente/servidor de acordo com um protocolo.

O foco do trabalho não é a interface de utilizador, portanto a prioridade é a implementação das funcionalidades do ponto de vista lógico devendo a interface de utilizador ser tão simples quanto possível.

O servidor deverá 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, enquanto os clientes poderão inserir, e ler mensagens, e alterar os dados na sua conta/perfil;

Os clientes podem funcionar em dois modos: o **modo registado** e **não registado**, em qualquer um deles é obrigatório ter um nickname/nome definido (que funcionará como um username vísivel e que deve ser único no servidor).

No **modo não registado**, o cliente funciona apenas no canal *default,* enquanto no modo registado inicia no canal preferido desse utilizador, que o próprio utilizador pode alterar.

Os utilizadores **registados** podem ter a categoria especial de operador que permite gerir canais e gerir outros utilizadores.

1.1 Protocolo de comunicação

A comunicação entre clientes e servidor deve obedecer a um protocolo de texto simples (*plain text single line protocol*).

Cada mensagem tem um comprimento máximo de 512 bytes, sendo iniciada por um comando com 4 caracteres seguida de um ou mais parâmetros separados por espaços em branco.

Uma mensagem pode dar origem (ou não) a uma resposta do servidor.

(1. Continuação)

1.2 Comandos pretendidos para comunicação e definições de registo

Comandos de utilizador (todos os utilizadores):

NICK <nome> - Atribui ou muda o nome/nickname com um máximo de 9 caracteres.

MSSG <mensagem> - Envia uma mensagem para o canal ativo (a mensagem re-enviada pelo servidor deve ser composta pela mensagem original precedida da identificação de quem a enviou).

Comandos de utilizador registado (pode ser operador ou não):

PASS <password> - Autenticação de um utilizador registado.

JOIN <canal> - Muda o canal ativo.

LIST - Lista os canais existentes.

WHOS - Lista e mostra informação dos utilizadores conectados no canal.

Comandos de Operador:

KICK <nome> - Expulsa o utilizador (remove da lista dos utilizadores registados).

REGS <nome ><password> - Regista o utilizador (inclui na lista dos utilizadores registados).

OPER <nome> - Promove o utilizador (registado) à categoria de operador.

QUIT -Desiste de ser operador.

Mensagens do servidor

RPLY código (código de resposta) / [listas] - Respostas a comandos.

MSSG "origem nome/categoria:> mensagem" - Outras mensagens.

2. Decisões tomadas no desenvolvimento do trabalho

- O grupo pensou inicialmente, para chegar ao objetivo final, definir uma estrutura (**'list_c'**) para gerir a informação de cada cliente.

- Após definida a estrutura, foram implementados os "esqueletos" de servidor e cliente para que fosse possível uma ligação e possível comunicação entre vários clientes e o servidor. Após diversas tentativas e alguns erros no desenvolvimento dos mesmos, e depois de alguma pesquisa e estudo dos "esqueletos" fornecidos no moodle (que serviram para resolver exercícios durante o ano nas aulas práticas e que nos foram muito úteis no trabalho final), o grupo conseguiu estabelecer ligações entre os mesmos com sucesso.

- Depois disso, o grupo focou-se no protocolo definido e nas funcionalidades que cada comando deveria executar, focando-se muito mais na implementação do servidor do que na implementação do cliente (como era pretendido).

- Os primeiros comandos a serem implementados foram então **NICK** e **MSSG** pois era a base necessária para inicializar um cliente e para que este pudesse executar qualquer outro comando. Como auxiliar foi implementada uma função (*'**RemoveFirst**'*) que remove a primeira string introduzida (o comando a executar) e devolve apenas a mensagem a enviar aos outros clientes.

- Após implementados com sucesso, o grupo decidiu seguir para os comandos de registo e autenticação (**REGS** e **PASS**), que também foram devidamente implementados após muitos erros e debug's.

2. Decisões tomadas no desenvolvimento do trabalho (Continuação)

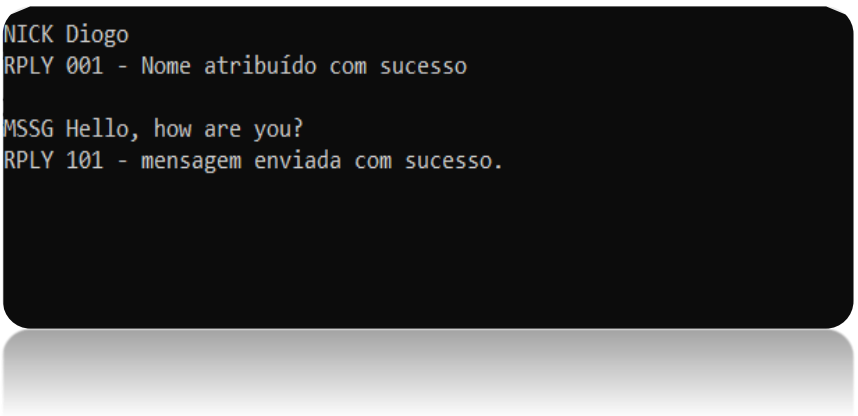
- Os últimos comandos a serem implementados foram então KICK, OPER E QUIT, por terem funcionalidades que dependiam de comandos anteriores e que só podiam ser utilizados depois destes serem inseridos pelo utilizador.

- Foi também implementado no final, o comando WHOS, que lista e mostra a informação dos utilizadores conectados no canal.

- Ficaram por implementar os comandos JOIN e LIST devido a uma falta de noção do que deverá ser definido como 'canal' e o também código previsto para correção de erros.

- Em contrapartida, para além dos comandos previstos no início do trabalho, foram implementados com sucesso, os comandos INFO (para o utilizador visualizar a sua informação) e EXIT (para o utilizador sair da sessão).

- Concluindo, achamos que foram atingidos pontos muito importantes do trabalho e foram implementadas funcionalidades essenciais (nunca trabalhadas antes desta cadeira) para a execução e um bom funcionamento da interface pedida no trabalho prático final.



3. Opções necessárias para compilação e teste do software

Para compilação do trabalho é apenas necessário o utilizador compilar o ficheiro 'server.c' usando:

- **make server**

Em seguida, o RUN é feito com:

- **./server <port_number> Ex. ./server 5555**

  Sendo este inicializado com:
  - **ip: 0.0.0.0 / 127.0.0.1 (default)**
  - **port: <port_number>**

Após inicializado o servidor, o utilizador deve passar para a compilação do ficheiro 'client.c' usando:

- **make client**

Em seguida, o RUN é feito com:

- **./client <ip_adress> <port_number> Ex: ./client 0.0.0.0 5555**

  De forma a este, ser inicializado no servidor.