Instituto Superior de Engenharia

Politécnico de Coimbra

Relatório do Trabalho Prático de Sistemas Operativos

Miguel Ferreira Cardoso (2022143267)
Rafael Filipe Rodrigues Pereira (2022150534)

Licenciatura em Engenharia Informática

Departamento de Engenharia Informática e Sistemas

Instituto Superior de Engenharia de Coimbra

Coimbra, 21 de outubro, 2024

Índice

Índice de Figuras

1 Introdução

Este trabalho prático consiste em desenvolver uma plataforma de envio e receção de mensagens curtas, organizadas por tópicos. Para isso foram desenvolvidos dois programas. O *manager* serve para gerir o envio e a receção de mensagens enviadas pelos utilizadores, gerir os tópicos existentes e também gerir os utilizadores da plataforma. O *feed* serve para interagir com a plataforma, enviar mensagens para tópicos, comunicando com outros utilizadores.

2 Estrutura do trabalho

O trabalho foi dividido em múltiplos ficheiros, cada um com o foco de gerir separadamente os aspetos do mesmo.

Um aspeto importante no projeto é a forma como um utilizador se conecta ao *manager*. Para estabelecerem conexão, verificando previamente se já existe um utilizador com o mesmo nome, é criada uma pipe chamada "tryConnect". Inicialmente o *manager* recebe um pedido através da pipe global. Assim que recebe esse pedido cria a pipe tryConnect e o utilizador passa a receber todo o feedback através dessa pipe até saber se pode estabelecer conexão. Assim que sabe o resultado, ambos fecham o fd e o *manager* elimina a pipe.

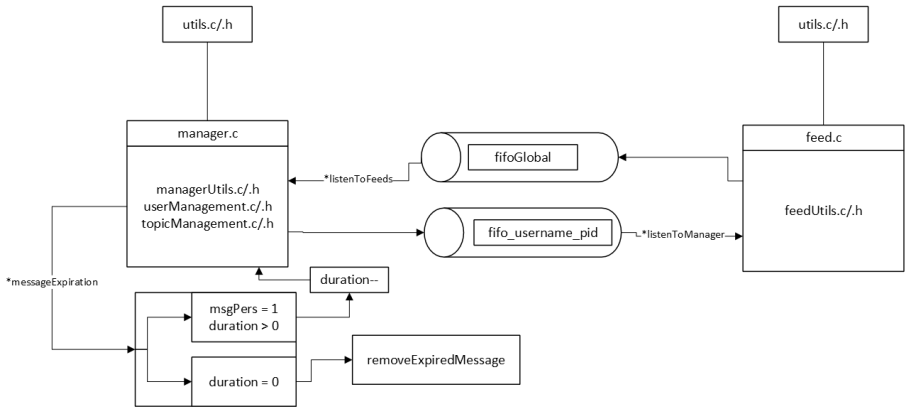

Figura 1: Estrutura Base

2.1 manager

O *manager* é responsável por gerir os utilizadores, os tópicos criados e as mensagens. Para tal, foi repartido em seis ficheiros para facilitar a sua construção e funcionamento.

Comandos implementados no *manager*:

- **users** - Mostra os utilizadores ativos.
- **remove <username>** - Remove um utilizador específico.
- **topics** - Mostra os tópicos existentes.
- **lock <topic>** - Bloqueia um tópico, sendo assim impossível enviar mensagens para este.
- **unlock <topic>** - Desbloqueia o tópico.
- **show <topic>** - Mostra todas as mensagens persistentes no tópico.
- **close** - Encerra o *manager* e todos os utilizadores.

2.1.1 managerUtils.c/.h

Este ficheiro contém 4 funções para o funcionamento básico do *manager*.

- ***void* handleCommand** – Serve para ler e executar os comandos do *manager*.
- ***void* *listenToFeeds** – É usada numa *thread*. Sempre que um utilizador faz um pedido, como mostrar os tópicos existentes ou se desconecta, esta função processa o pedido e realiza as ações necessárias (*2.2 - Exemplo de comunicação*).
- ***void* listUsers** – Função para mostrar a lista de utilizadores ativos.
- ***void* shutdownManager** – Encerra o *manager*, trata de desconectar os utilizadores ativos, eliminar os tópicos existentes, etc.

2.1.2 userManagement.c/.h

Este conjunto de ficheiros possui todas as funções necessárias para a administração de utilizadores e a estrutura que define um utilizador.

***User struct*:**

- ***char* username[50]** – Nome de utilizador.
- ***int* pid** – ID do processo associado ao utilizador.
- ***char* userPipe[50]** – Nome do pipe do utilizador.
- ***char* subscriptions[MAX_TOPICS][MAX_TOPIC_NAME]** – Tópicos nos quais o utilizador está inscrito.
- ***int* sub_count** – Número de tópicos inscrito.

Estes ficheiros contêm as funções **findUser*, *userExists*, *addUser*, *removeUser*, *sendResponseToUser*, *subscribeUserToTopic*, *createPipe* e *cleanupPipe*. Todas estas funções servem para administrar os utilizadores. Dado que o *manager* deve gerir todas as pipes no projeto, as funções para criar e limpar as pipes são aqui incluídas, pois fazem parte da gestão dos utilizadores (comunicação).

Além das funções anteriores, existem ainda sete *handles* para lidar com os pedidos do utilizador. Na comunicação entre o *manager* e o *feed* são utilizados tipos de mensagens e respostas (*2.3.1 - Tipos de mensagem*). Sempre que o *manager* recebe um pedido de um utilizador, a função *processMessage* identifica o tipo da mensagem através do campo *type* e chama um dos *handles* para concretizar o pedido (*2.2 - Exemplo de comunicação*).

2.1.3 topicManagement.c/.h

Neste conjunto estão todas as funções para gerir os tópicos, as mensagens enviadas para os mesmos e a estrutura que define um tópico. Temos as funções básicas para a gestão dos tópicos, ou seja, uma função para criar tópicos, para os eliminar, para bloquear o envio de mensagens, desbloquear o envio de mensagens, etc. No entanto, há um conjunto de funções menos genéricas e importantes para o objetivo final do projeto.

Topic Struct

- ***char*** **name[MAX_TOPIC_NAME]** – Nome do tópico.
- ***int*** **is_locked** – Ou está bloqueado (1), ou está desbloqueado (0).
- ***int*** **msg_count** – Número de mensagens existentes no tópico (persistentes).
- ***Message*** **messages[MAX_MESSAGES_PER_TOPIC]** – Array de mensagens no tópico.

Funções mais importantes:

- ***void*** **broadcastToSubscribers** – Envia as mensagens para todos os utilizadores inscritos no tópico.
- ***void*** **addMessageToTopic** – Adiciona mensagens persistentes ao tópico.
- ***void*** **checkAndDeleteEmptyTopic** – Verifica se o tópico está vazio (mensages e utilizadores inscritos) e elimina o tópico se assim for.
- ***void*** ***messageExpiration** – Usada numa *thread* para decrementar o tempo das mensagens persistentes.
- ***void*** **notifyNewSubscriber** – Envia as mensagens persistentes para os utilizadores que se inscrevem no tópico após o envio destas.
- ***void*** **removeExpiredMessage** – Remove mensagens persistentes quando o tempo destas chega ao fim.

2.2 feed

O *feed* é responsável por tratar da interação do cliente com o *manager*. O seu funcionamento baseia-se no envio de comandos para o *manager*, sendo estes processados pelo *manager* e enviada uma resposta que será mostrada no *feed*. Isto é possível devido à implementação de tipos de mensagens (*2.3.1 - Tipos de mensagem*). Por exemplo, o comando *list* no *feed* envia uma mensagem do tipo *list*. Quando o *manager* recebe a mensagem e identifica o tipo de mensagem, processa o output e envia uma resposta ao *feed* com as informações necessárias.

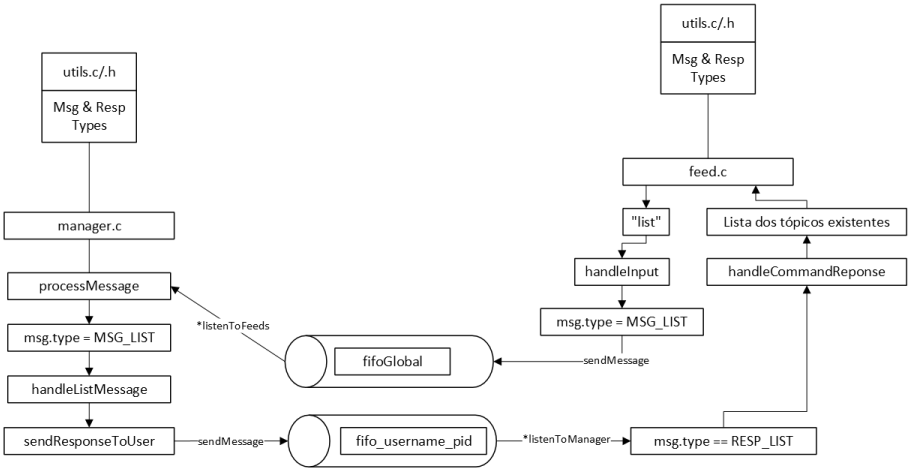

Figura 2: Exemplo de comunicação

2.2.1 feedUtils.c/.h

Nestes ficheiros estão as funções utilizadas no *feed* e a estrutura *FeedContext*. Esta estrutura é muito importante para o projeto, pois facilita bastante a lógica do *feed*. A estrutura *FeedContext* contém todas as informações essenciais para o *feed* funcionar. Contém os *file descriptors* para a comunicação e o nome de utilizador para se identificar quando envia uma mensagem para o *manager*.

***FeedContext Struct*:**

- ***char* username[MAX_USERNAME]** – Nome do utilizador.
- ***char* pipe_name[50]** – Nome da pipe do *feed*.
- ***int* managerPipe** – FD do *manager*.
- ***int* responsePipe** – FD do *feed*.
- ***int* running** – Indica se o *feed* está correr ou não.

Funções:

- ***void* initFeed** – Inicia o *feed*. Tenta estabelecer conexão e recebe uma resposta de sucesso ou erro (insucesso) do *manager*. Envia toda a informação necessária para tal.

- ***void* handle_input** – Lê todos os comandos introduzidos pelo utilizador. Consequentemente processa esses comandos e faz pedidos ao *manager*.
- ***void* shutdownFeed** – Encerra o *feed*. Envia uma mensagem de desconexão para o manager e encerra.
- ***void* *listen_manager** – Usada numa *thread* para receber mensagens do *manager*. Processa remoção por parte do *manager* (remove user), ou uma mensagem normal. Para outro tipo de mensagem, a próxima função trata de tal.
- ***void* handleCommandResponse** – Processa determinadas mensagens do *manager*, como mensagens de erro ou sucesso.

2.3 utils.c/.h

Aqui estão as funções, estruturas e definições comuns tanto ao *feed* quanto ao *manager*. As únicas funções existentes são *readInput* e *sendMessage*. Temos presente a estrutura das mensagens e um dos aspetos mais importantes do projeto, os tipos de mensagem.

2.3.1 Tipos de mensagem

Os "tipos" definem o tipo da mensagem, isto é, se é uma mensagem normal, se é para subscrever ou desinscrever a um tópico, se é uma mensagem de erro ou sucesso num pedido, etc. O seu uso facilita bastante a comunicação entre os processos e a lógica para processar comandos. Por exemplo, em vez de utilizar sinais, quando o *manager* quer remover um utilizador envia uma mensagem do tipo *RESP_BYE_BYE*. Quando o *feed* recebe essa mensagem executa a função *shutdownFeed*, a qual envia uma mensagem a avisar que se desconectou. Então o *feed* encerra e o *manager* remove o utilizador da sua memória, terminando assim a remoção do utilizador por parte do *manager*.

Message struct:

- ***char* topic[MAX_TOPIC_NAME]** – Nome do tópico.
- ***char* sender[MAX_USERNAME]** – Emissor da mensagem (feed (username) ou manager).
- ***char* content[MAX_MSG_LEN]** – Corpo da mensagem.
- ***int* duration** – Duração da mensagem.
- ***int* pers** – Se a duração for maior que 0 fica a 1, caso contrário 0.
- ***int* type** – Tipo de mensagem (sucesso, erro, message, etc).

Ao longo do código é possível ver o pid do feed ser colocado no campo duration quando está a ser estabelecida a conexão entre o feed e o manager. Dado que para esse contexto vários campos da estrutura Message não são usados, esse campo foi aproveitado para passar o pid de feed para o manager.

3 Conclusão

O projeto cumpre o objetivo de permitir o envio e receção de mensagens curtas, organizadas por tópicos. O *manager* gere todos os aspetos do sistema. O *feed* facilita a interação dos utilizadores com a plataforma, permitindo enviar mensagens e receber respostas.

A organização do código com a separação em vários ficheiros, cada par dedicado a diferentes aspetos do sistema, garantiu uma implementação clara e eficiente.