Instituto Superior de Engenharia

Politécnico de Coimbra

Trabalho Prático
Programação em C para UNIX

Sistemas Operativos

Alexandre Moreira
a2020144214@isec.pt
Rui Monteiro
a2022148943@isec.pt

Licenciatura em Engenharia Informática
ISEC

Coimbra, 01 de Novembro 2023

Índice

Capítulo 1

Introdução

O trabalho prático proposto na disciplina de **Sistemas Operativos** tem como objetivo o desenvolvimento de uma plataforma de mensagens curtas organizada por tópicos. O projeto consiste em dois programas principais: o **manager**, responsável pela gestão centralizada da plataforma, e o **feed**, utilizado pelos utilizadores para enviar e receber mensagens. A comunicação entre esses programas é realizada em um ambiente Unix, utilizando técnicas de programação em C e mecanismos do sistema operacional estudados durante as aulas.

Este trabalho foca-se na aplicação prática de conceitos essenciais, como sincronização de threads, comunicação por meio de *named pipes*, e manipulação eficiente de recursos do sistema.

Os programas suportam duas categorias de utilizadores: clientes e administradores. Os **clientes** interagem por meio do programa 'feed' para: - Registrar-se na plataforma; - Enviar mensagens para tópicos específicos; - Subscrever-se a tópicos para receber mensagens em tempo real.

O **administrador**, utilizando o programa 'manager', é responsável por: - Controlar os tópicos e mensagens; - Listar utilizadores ativos; - Bloquear ou desbloquear tópicos; - Encerrar a plataforma de forma ordenada.

O sistema também lida com mensagens persistentes, que são armazenadas temporariamente no servidor até expirarem, e não persistentes, que são entregues apenas aos utilizadores conectados no momento do envio.

Este relatório apresenta a arquitetura do sistema, a estrutura de implementação e as principais decisões tomadas no desenvolvimento da solução, com base nos requisitos funcionais e técnicos descritos no enunciado do trabalho.

Capítulo 2

Arquitetura

A arquitetura do sistema foi projetada para atender aos requisitos funcionais e técnicos do trabalho prático, utilizando comunicação via *named pipes* e sincronização eficiente para evitar condições de corrida. O sistema é composto por dois programas principais: o **manager** e o **feed**, que interagem para criar uma plataforma robusta de mensagens organizadas por tópicos. A seguir, são detalhados os principais componentes e suas interações.

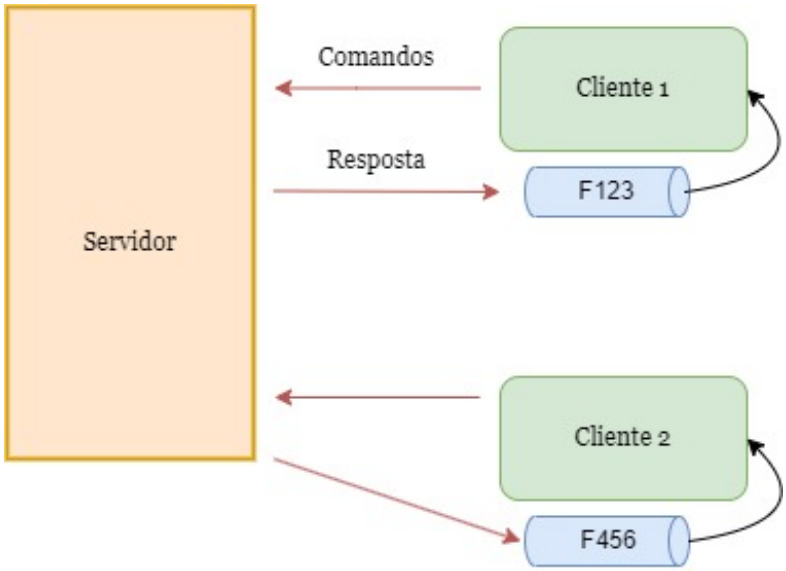

Figura 2.1: Esquema da Arquitetura

2.1 Threads

No programa **manager**, as **threads** são utilizadas para permitir a execução simultânea de múltiplas tarefas, garantindo que o sistema possa lidar com diversas operações de forma eficiente e sem bloqueios desnecessários. A utilização de **threads** possibilita o processamento concorrente de:

- Comandos do administrador (interação com o stdin).
- Mensagens recebidas dos utilizadores (clientes que enviam comandos via named pipes).
- Atualização do tempo de vida das mensagens persistentes (verificação contínua e remoção de mensagens expiradas).

2.1.1 Implementação

Funções das Threads

- **thread admin**: Lida com os comandos introduzidos pelo administrador, como listar utilizadores ou bloquear tópicos.
- **thread le pipe**: Lê mensagens dos pipes enviados pelos programas feed (clientes).
- **thread monitorar mensagens|**: Atualiza periodicamente as mensagens persistentes, removendo as que já expiraram.

Sincronização entre Threads

Para proteger o acesso a recursos compartilhados, como listas de utilizadores, tópicos e mensagens, são utilizados mutexes (`pthread_mutex_t`). As threads bloqueiam os recursos críticos usando `pthread_mutex_lock()` e libertam com `pthread_mutex_unlock()`.

Capítulo 3

Funcionamento

O sistema implementa uma plataforma de mensagens organizada por tópicos e é composto por dois programas principais: manager e feed. Abaixo é descrito o funcionamento do sistema em detalhe, incluindo a comunicação e a gestão de recursos

3.1 Inicialização do Sistema

- O programa **manager** é iniciado e entra em modo de espera para gerenciar a plataforma.

- O **manager** cria um **FIFO global** (named pipe) que será utilizado para a receção de comandos provenientes dos clientes.

- Cada cliente, ao ser iniciado através do programa feed, comunica-se com o manager por meio deste **FIFO global**.

- **Validações** são feitas para garantir que o nome do utilizador seja único e que o número máximo de utilizadores não tenha sido atingido.

3.2 Registo e Comunicação entre Clientes e Manager

3.2.1 Registo do Cliente

- Cada cliente, ao iniciar, envia o seu **nome** e **PID** ao manager para registo.
- O manager valida os dados e cria um **FIFO** exclusivo para cada cliente, nomeado como `f<PID>`. Esse FIFO será usado para comunicação direta entre o manager e o cliente.

3.2.2 Comandos do Cliente

O cliente pode enviar comandos ao **manager** através do **FIFO global** (ex.: envio de mensagens, subscrição de tópicos, cancelamento de subscrições). O **manager** processa os comandos recebidos e responde ao cliente pelo **FIFO exclusivo**.

3.2.3 Distribuição de Mensagens

Quando um cliente envia uma mensagem para um tópico:

- O **manager** verifica se o tópico existe ou cria um novo tópico, se necessário.
- A mensagem é distribuída para todos os **clientes subscritos** ao tópico em questão.

Para mensagens **persistentes**, o **manager** armazena temporariamente a mensagem na memória, permitindo que novos subscritores recebam mensagens antigas dentro do tempo de vida definido.

3.2.4 Sincronização e Threads no Manager

Quando o **manager** é iniciado, ele cria várias **threads** para gerenciar simultaneamente diferentes funcionalidades:

- **Thread de Comandos do Administrador**: Responsável por ler os comandos do administrador (ex.: listar utilizadores, bloquear tópicos).
- **Thread de Leitura de Mensagens**: Lê os comandos enviados pelos clientes através do FIFO global.
- **Thread de Monitoramento de Mensagens Persistentes**: Verifica continuamente o tempo de vida das mensagens persistentes e remove as expiradas.

3.2.5 Processo de Comunicação com Tópicos

1. **Subscrição de Tópicos**:
   - Um cliente pode subscrever-se a um tópico usando o comando `subscribe <topico>`. Se o tópico não existir, ele é criado.
   - Quando um tópico é subscrito, todas as mensagens persistentes associadas a ele são imediatamente enviadas ao cliente.
2. **Envio de Mensagens**:
   - O cliente envia uma mensagem com o comando `msg <topico> <duração> <mensagem>`.
   - O manager processa a mensagem e a distribui aos clientes subscritos.
   - Se a mensagem tiver **duração** > 0, ela é armazenada como uma **mensagem persistente**.
3. **Receção de Mensagens**:
   - Os clientes recebem mensagens enviadas ao FIFO `cli_<PID>` correspondente.
   - Esse FIFO é gerenciado pelo **manager**, que encaminha mensagens de tópicos subscritos.

3.2.6 Encerramento e Gestão de Recursos

1. **Saída de Clientes:**
    - Um cliente pode sair da plataforma com o comando `exit`. O **manager** remove o utilizador e fecha o FIFO associado.
    - Se o cliente estiver subscrito a tópicos, as subscrições são canceladas.
2. **Encerramento do Manager:**
    - O administrador pode encerrar a plataforma usando o comando `close`.
    - Antes de finalizar, o **manager** realiza as seguintes tarefas:
        - Armazena mensagens persistentes no ficheiro definido.
        - Fecha e elimina todos os FIFOs criados.
    - Todos os clientes conectados são notificados do encerramento e finalizam os seus processos de forma ordeira.

Capítulo 4

Conclusão

4.1 Conclusão

Neste trabalho, desenvolvemos uma plataforma de envio e receção de mensagens organizada por tópicos, composta por dois programas principais: o **manager**, responsável pela gestão centralizada de utilizadores, tópicos e mensagens, e o **feed**, utilizado pelos clientes para interagir com a plataforma.

A implementação envolveu o uso de mecanismos fundamentais do sistema operativo Unix, como:

- **Named Pipes (FIFOs)**: Utilizados para a comunicação eficiente entre processos, garantindo a troca de mensagens entre o *manager* e os *feeds*.
- **Threads**: Implementadas no programa *manager* para gerenciar simultaneamente a receção de mensagens, monitorização do tempo de vida das mensagens persistentes e execução de comandos do administrador.
- **Sincronização com Mutexes**: Garantindo o acesso seguro às estruturas de dados compartilhadas, como listas de utilizadores, tópicos e mensagens.
- **Persistência de Dados**: Mensagens persistentes foram armazenadas em ficheiros de texto, possibilitando a recuperação após o reinício do sistema.

Durante o desenvolvimento, enfrentamos desafios relacionados com a sincronização de dados e o gerenciamento concorrente das threads, problemas que foram resolvidos com o uso adequado de mutexes e boas práticas de programação concorrente.

O sistema cumpre os requisitos estabelecidos, oferecendo:

- Registo e validação de utilizadores.
- Criação automática e gestão de tópicos.
- Envio de mensagens persistentes e não-persistentes.
- Subscrição e cancelamento de subscrições de tópicos.
- Funcionalidades administrativas, como listagem de utilizadores, bloqueio de tópicos e encerramento da plataforma.

Por fim, este projeto permitiu aplicar na prática conceitos essenciais de comunicação, concorrência e manipulação eficiente de recursos do sistema Unix.