Instituto Superior de Engenharia

Politécnico de Coimbra

DEPARTAMENTO DE / DEPARTMENT OF
INFORMÁTICA E SISTEMAS

Trabalho Prático de Sistemas Operativos

Autores / Authors

Diogo Marques Silva

João Pedro Vila Pomar

Unidade Curricular

Sistemas Operativos

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, dezembro 2024

ÍNDICE

1 INTRODUÇÃO

Este projeto tem como objetivo a implementação de uma plataforma de mensagens, baseada em tópicos, em linguagem C e com mecanismos de comunicação do sistema operativo Unix. A plataforma consiste num servidor "*manager*" que gere utilizadores, tópicos e mensagens, e um cliente denominado "*feed*" que permite aos utilizadores interagirem com a plataforma, enviando mensagens e subscrevendo tópicos. A comunicação entre o *feed* e o manager é intermediada por "*named pipes*", que garantem a sincronização e a troca de mensagens no ambiente.

2 ARQUITETURA DO SISTEMA

A arquitetura do sistema foi desenhada para ser modular e escalável, dividindo as responsabilidades entre os dois componentes principais, baseada numa comunicação entre processos utilizando FIFOs *(named pipes),* onde a comunicação é feita entre o *manager* e os *feeds*.

2.1 Componentes Principais

O *manager* é responsável por gerir os utilizadores conectados e os tópicos existentes, as mensagens que o utilizador envia para um determinado tópico são distribuídas aos outros utilizadores subscritos pelo manager. O manager inclui funcionalidades administrativas para bloquear e desbloquear tópicos, assim como remover utilizadores quando pretender.

No lado do *feed*, este funciona como interface onde o utilizador pode interagir com a plataforma, é permitido ao utilizador executar diversas ações, depois de se apresentar (introduzindo o nome), como subscrever tópicos, enviar mensagens, consultar tópicos ativos e subscritos por ele próprio. Nessa interface, o utilizador

recebe mensagens de outros utilizadores aquando da sua subscrição ao mesmo tópico.

2.2 Comunicação entre processos

Como já foi referido anteriormente, o *feed* e o *manager* utilizam *named pipes* para comunicarem entre si. O manager utiliza um *pipe* principal (PIPE_MANAGER) para receber as mensagens dos clientes. Cada cliente, a fim de receber as mensagens do manager cria um *pipe* exclusivo, ou seja, único. Para fazer com que os *pipes* de cada feed sejam únicos foram aproveitados os *pid* que corresponde a um só processo. A comunicação entre estes é baseada no envio de estruturas

3 ESTRUTURAS DE IMPLEMENTAÇÃO

3.1 . Estruturas de Dados

"MensagemManager" é a estrutura central para a troca de informações entre *feed* e manager, e incluem o tipo de mensagem, nome de utilizador, nome do tópico, conteúdo da mensagem, e duração (para as mensagens persistentes).

Para representar cada tópico, usamos a estrutura "TopicoManager" que contém a lista de subscritores, as mensagens persistentes e o estado do tópico, que pode estar bloqueado ou desbloqueado.

No tópico "ManagerData", está presente a estrutura global do manager, contendo listas de utilizadores e tópicos, bem como *mutexes* para sincronização.

Para armazenar informações sobre os pipes e tópicos subscritos, é usada a estrutura "FeedData", que é a estrutura local do *feed*.

4 JUSTIFICAÇÃO DAS DECISÕES E MODELOS SEGUIDOS

4.1 Threads e Mutexes e Sinais

No *feed*, foram usadas *threads* para permitir a execução concorrente de diferentes partes do programa:

- Receção de Mensagens: uma *thread* foi criada para ler e processar mensagens do manager.
- Tratamento de Comandos do Utilizador: a *thread* principal do programa é responsável por tratar os comandos do utilizador.

Os sinais foram usados para permitir que uma *thread* notifique outra sobre a necessidade de encerrar o programa, mais especificamente quando um *feed* tenta entrar na plataforma, mas é recusada a sua entrada, quando o manager o remove ou quando o manager é encerrado:

- Sinal SIGUSR1: Usado para notificar a *thread* de comandos do utilizador para encerrar o programa quando o manager envia uma mensagem de remoção do utilizador.

No manager foram usadas *threads* para permitir a execução concorrente de diferentes partes do programa, como o tratamento dos comandos do manager, a receção de mensagens e a decrementação do tempo de vida das mensagens persistentes. Mutexes foram usados para garantir a exclusão mútua ao aceder a recursos compartilhados, como o acesso a subscritores e o tratamento de mensagens, garantindo a integridade dos dados. Estes mecanismos permitem que o programa funcione de forma eficiente e segura em um ambiente concorrente, melhorando a capacidade de resposta e a robustez do sistema.

5 DESAFIOS ENCONTRADOS

Durante o desenvolvimento do projeto, surgiram diversos desafios que aumentaram a complexidade do trabalho. Entre eles, destacam-se a correção de "bugs" relacionados à exibição das respostas de certos comandos executados pelo utilizador e pelo manager. Outro desafio significativo foi garantir que os utilizadores subscritos nos tópicos recebessem corretamente as mensagens enviadas por outros utilizadores. Além disso, a implementação de mecanismos robustos de tratamento de erros foi fundamental, especialmente para lidar com falhas na criação de *threads* e na abertura de *pipes*, assegurando a estabilidade e o bom funcionamento do código.

6 CONCLUSÃO

Em suma, o desenvolvimento deste projeto exigiu a resolução de diversos desafios e a implementação de soluções complexas para garantir a funcionalidade e estabilidade do sistema, respeitando os objetivos requeridos, como as funcionalidades de todos os comandos a utilizar pelo manager e pelo *feed*. Este trabalho permitiu obter uma visão abrangente sobre a construção de sistemas distribuídos e a importância de ter uma arquitetura bem planejada, de forma que toda a plataforma funcione sem erros e com eficiência.

Instituto Superior de Engenharia
Politécnico de Coimbra