

Sistemas Operativos

Ano letivo de 2024 / 2025

Relatório do Trabalho Prático

Autores:

2022147149 – Bruno Tiago Ferreira Martins

2022113359 – Nuno José Soares Cerejeira

Índice

1. Introdução	3
2. Arquitetura do Sistema	4
3. Descrição da Implementação	7
4. Comandos e Interações	10
5. Resultados e Validação	11
6. Conclusão	12

1. Introdução

A crescente complexidade dos sistemas operativos modernos exige uma compreensão aprofundada dos seus mecanismos fundamentais, especialmente no que diz respeito à comunicação entre processos e à gestão eficiente de recursos. Este trabalho prático foi desenvolvido com o objetivo de proporcionar uma experiência prática e aplicada no desenvolvimento de uma plataforma de mensagens organizada por tópicos, utilizando a linguagem C no ambiente Unix.

Objetivo Geral: Desenvolver uma plataforma funcional e robusta, composta por dois programas principais: Manager e Feed. Esta plataforma permite o envio, receção e gestão de mensagens, explorando conceitos como comunicação por *pipes*, sincronização de threads, e persistência de dados.

Objetivos Específicos:

Implementar um sistema que permita a comunicação eficaz entre múltiplos utilizadores através de tópicos.

Gerir mensagens persistentes com tempos de vida definidos.

Garantir a integridade dos dados e a sincronização entre threads em cenários de alta concorrência.

Explorar a recuperação de dados persistentes após interrupções do sistema.

Estrutura da Plataforma: A solução é composta por dois programas:

Manager: Responsável pela gestão centralizada dos tópicos, utilizadores e mensagens. Atua como o núcleo do sistema.

Feed: Interface do utilizador que permite enviar mensagens, subscrever tópicos e visualizar mensagens recebidas. Cada instância do Feed representa um utilizador.

Esta abordagem promove uma separação clara de responsabilidades e garante escalabilidade e modularidade na implementação.

2. Arquitetura do Sistema

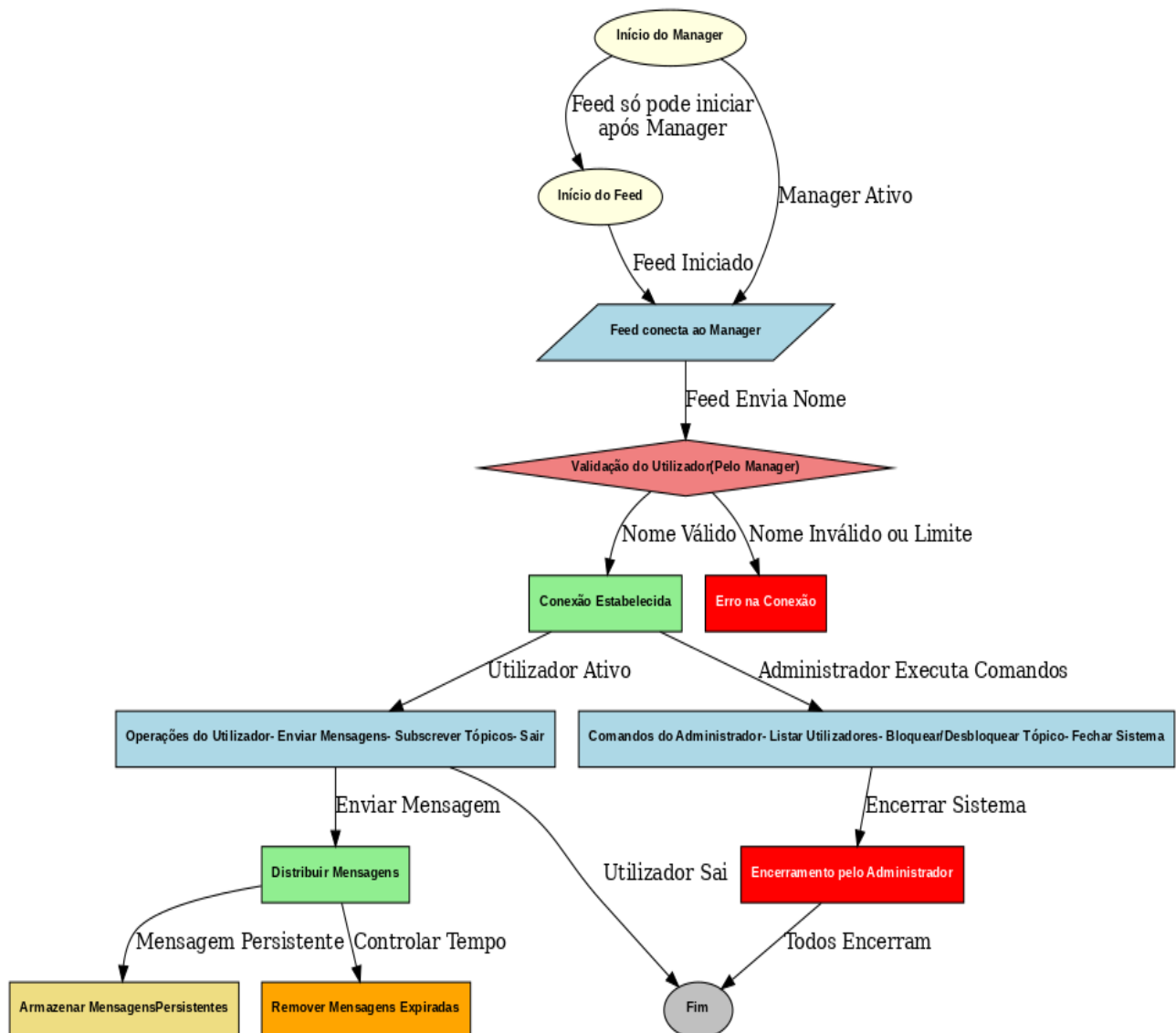


Figura 1 - Fluxograma do projeto

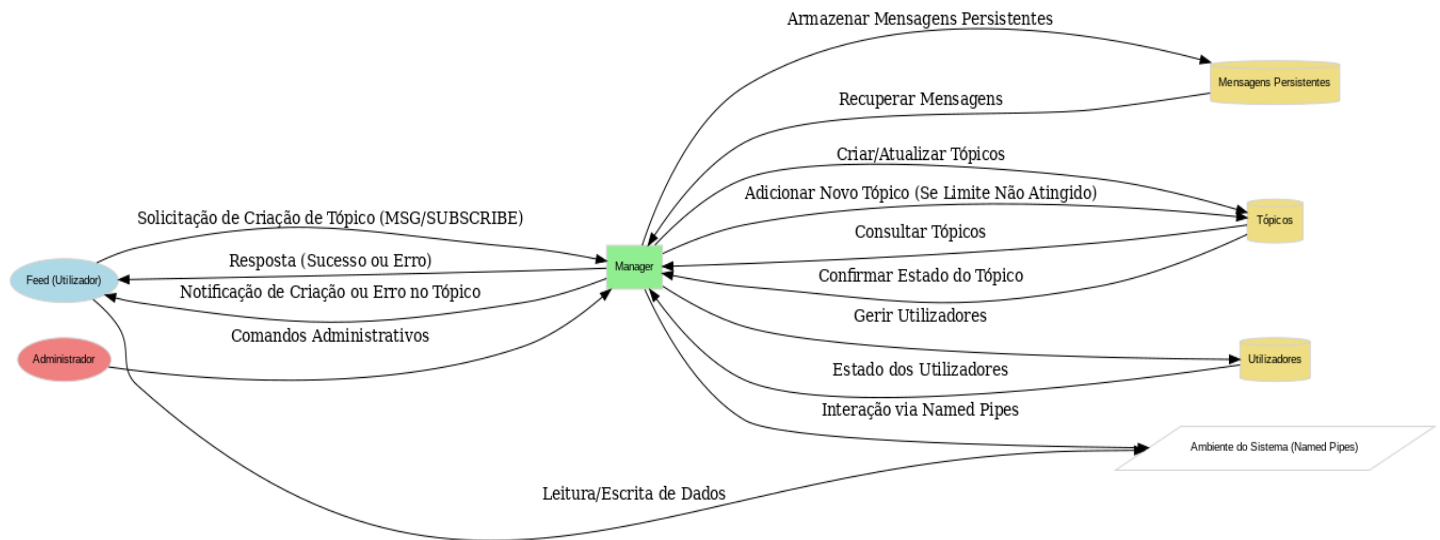


Figura 2 - Diagrama de Fluxo de dados do projeto

Componentes Principais

Componente	Descrição
Manager	Garante a distribuição das mensagens, cria tópicos automaticamente e gere os utilizadores e mensagens persistentes.
Feed	Programa cliente que permite aos utilizadores subscrever tópicos, enviar mensagens e visualizar o conteúdo recebido.

Mecanismos de Comunicação

A comunicação entre os programas Feed e Manager ocorre através de *named pipes*. Cada utilizador do sistema tem um pipe exclusivo para troca de mensagens. A tabela abaixo apresenta os detalhes:

Elemento	Descrição
<i>Named Pipes</i>	Usados para troca de mensagens entre Feed e Manager.
<i>Pipe Exclusivo</i>	Cada utilizador tem um pipe dedicado para comunicação.
<i>Distribuição</i>	O Manager distribui mensagens de tópicos aos Feeds subscritos.
Comunicação Bidirecional	Permite envio e receção simultânea de mensagens entre utilizadores via Manager.

Estrutura do Fluxo de Dados

O sistema foi projetado para que o fluxo de dados seja gerido eficientemente, como detalhado abaixo:

1. **Envio de Mensagens:**
 - O Feed envia uma mensagem ao Manager especificando o tópico e o tempo de vida
 - Se o tópico não existir (e não tiver sido ultrapassado os limites), o Manager cria-o automaticamente.
2. **Distribuição:**
 - O Manager verifica os utilizadores subscritos ao tópico e distribui a mensagem imediatamente.
 - Mensagens persistentes são armazenadas até expirarem ou serem carregadas por novos subscritores.
3. **Receção:**
 - Os Feeds recebem as mensagens dos tópicos subscritos em tempo real.

Sincronização e Recursos

Recurso	Mecanismo Utilizado
Sincronização entre threads	pthread_mutex_t para proteção de secções críticas.
Gestão de mensagens persistentes	Ficheiros de texto, definidos pela variável de ambiente MSG_FICH.
Controlo de acessos concorrentes	Uso de <i>locks</i> para evitar conflitos entre threads.

Gestão de Estados

Estado	Descrição
Tópicos Bloqueados	Os tópicos marcados como "bloqueados" não aceitam novas mensagens, mas continuam disponíveis para leitura e subscrição.
Utilizadores Ativos	Apenas utilizadores autenticados podem interagir com a plataforma.

Esta arquitetura modular e sincronizada garante a escalabilidade e robustez do sistema ao lidar com múltiplos utilizadores simultâneos.

3. Descrição da Implementação

Estruturas de Dados

Estrutura	Descrição
-----------	-----------

Mensagem	Representa uma mensagem enviada, contendo informações como tópico, autor, conteúdo, e tempo de vida.
-----------------	--

User	Armazena os dados de um utilizador, incluindo tópicos subscritos e estado de conexão.
-------------	---

Topic	Contém o nome do tópico e informações sobre o estado (bloqueado/desbloqueado).
--------------	--

Estrutura Message em código:

```
typedef struct {
    TipoComando comando;           // Tipo de comando
    char topico[MAX_TOPICO_NOME];  // Nome do tópico
    char username[MAX_USERNAME];  // Nome do utilizador
    char pipe_name[50];           // Nome do pipe do feed
    int persistente;              // Indica se a mensagem é persistente
    int tempo_de_vida;            // Tempo de vida em segundos para mensagens persistentes
    size_t msg_len;               // Tamanho real da mensagem
    char mensagem[MAX_MSG_LEN];   // Conteúdo da mensagem
    int removida;                 // Flag para indicar se a mensagem foi removida
} Mensagem;
```

Gestão de Tópicos

Os tópicos são geridos no Manager e criados automaticamente caso ainda não existam. Exemplo de função para criar ou localizar um tópico, o feed ao enviar uma mensagem para um tópico, não fica automaticamente subscrito:

```
int procura_ou_cria_topico(char *nome_topico, char *pipe_name, int subscricao) {
    // Procurar se o tópico já existe
    for (int i = 0; i < num_topicos; i++) {
        if (strcmp(topicos[i].nome, nome_topico) == 0) {
            return i; // Retorna o índice do tópico existente
        }
    }

    // Verificar se o limite de tópicos foi atingido ao criar um novo tópico
    if (num_topicos >= MAX_TOPICOS) {
        return -1; // Indica que o limite foi atingido
    }

    // Criar novo tópico
    strcpy(topicos[num_topicos].nome, nome_topico);
    topicos[num_topicos].bloqueado = 0;

    return num_topicos++; // Retorna o índice do novo tópico
}
```

Distribuição de Mensagens

As mensagens são distribuídas pelo Manager para todos os utilizadores subscritos. Exemplo da função de distribuição:

```
void distribui_mensagem(Mensagem *msg) {

    printf("Distribuindo mensagem do tópico %s para os subscritores.\n", msg->topico);

    // Verifica se a mensagem é persistente e deve ser armazenada
    if (msg->persistente) {
        int topic_index, msg_index;

        // Procura um espaço disponível para armazenar a mensagem persistente
        if (procura_lugar_vazio_persistentes(msg->topico, &topic_index, &msg_index) == 0) {
            mensagens_persistentes[topic_index][msg_index] = *msg; // Armazena a mensagem
            mensagens_persistentes[topic_index][msg_index].removida = 0; // Marca como ativa
            printf("Mensagem persistente guardada no tópico %s\n", msg->topico);
        } else {

            printf("Erro: Não foi possível guardar a mensagem persistente (sem espaço)\n");
        }
    }

    // Verifica se o tópico associado está bloqueado
    for (int i = 0; i < num_topicos; i++) {
        if (strcmp(topicos[i].nome, msg->topico) == 0) { // Encontra o tópico correspondente
            if (topicos[i].bloqueado) {
                printf("O tópico %s está bloqueado. Mensagem não enviada.\n", msg->topico);

                // Cria uma mensagem de notificação para o remetente
                Mensagem msg_notificacao;
                msg_notificacao.comando = MSG;
                strcpy(msg_notificacao.mensagem, "A mensagem não foi enviada pois o tópico está bloqueado");
                strcpy(msg_notificacao.topico, msg->topico);
                strcpy(msg_notificacao.username, "Sistema");
                strcpy(msg_notificacao.pipe_name, msg->pipe_name); // Define o pipe de retorno
                // Envia a notificação para o remetente
                int fd = open(msg->pipe_name, O_WRONLY);
                if (fd != -1) {
                    write(fd, &msg_notificacao, sizeof(Mensagem));
                    close(fd);
                } else {
                    perror("Erro ao abrir pipe para enviar notificação");
                }
                return; // Interrompe a distribuição
            }
            break; // Tópico encontrado e não está bloqueado
        }
    }

    // Distribui a mensagem para todos os utilizadores inscritos no tópico
    for (int i = 0; i < num_utilizadores; i++) {
        if (utilizadores[i].conectado) { // Apenas para utilizadores conectados
            for (int j = 0; j < utilizadores[i].num_subscritos; j++) {
                if (strcmp(utilizadores[i].topicos_subscritos[j], msg->topico) == 0) {
                    // Encontra um utilizador subscrito no tópico
                    int fd = open(utilizadores[i].pipe_name, O_WRONLY); // Abre o pipe do utilizador
                    if (fd != -1) {
                        write(fd, msg, sizeof(Mensagem)); // Envia a mensagem
                        close(fd);
                        printf("Mensagem enviada para %s: %s\n", utilizadores[i].username, msg->mensagem);
                    } else {
                        perror("Erro ao abrir pipe para enviar mensagem");
                    }
                    break; // Envia para um utilizador e passa para o próximo
                }
            }
        }
    }
}
```

Formato das Mensagens Persistentes:

<nome do tópico> <username> <tempo de vida restante> <mensagem>

```
void guarda_mensagens_persistentes() {  
    // Obter o nome do ficheiro a partir da variável de ambiente  
    char *filename = getenv("MSG_FICH");  
    if (!filename) {  
        printf("Variável de ambiente MSG_FICH não definida\n");  
        return;  
    }  
  
    // Abrir o ficheiro em modo de escrita (criar ou truncar se necessário)  
    int file = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0666);  
    if (file == -1) {  
        perror("Erro ao abrir ficheiro de mensagens persistentes");  
        return;  
    }  
  
    char buffer[1024];  
    // Percorrer todos os tópicos e mensagens persistentes  
    for (int i = 0; i < num_topicos; i++) {  
        for (int j = 0; j < MAX_MSGS_PERSISTENTES; j++) {  
            // Apenas guardar mensagens não removidas e com tempo de vida válido  
            if (!mensagens_persistentes[i][j].removida &&  
                mensagens_persistentes[i][j].tempo_de_vida > 0) {  
                // Formatar os dados da mensagem para o ficheiro  
                int len = snprintf(  
                    buffer, sizeof(buffer), "%s %s %d %s\n",  
                    topicos[i].nome, // Nome do tópico  
                    mensagens_persistentes[i][j].username, // Autor da mensagem  
                    mensagens_persistentes[i][j].tempo_de_vida, // Tempo restante  
                    mensagens_persistentes[i][j].mensagem); // Conteúdo  
                // Escrever os dados formatados no ficheiro  
                write(file, buffer, len);  
            }  
        }  
    }  
    // Fechar o ficheiro após a escrita  
    close(file);  
}
```

Sincronização

O uso de `pthread_mutex_t` garante que múltiplas threads possam manipular recursos partilhados sem conflitos. Exemplo de bloqueio:

```
pthread_mutex_lock(&lock);
```

```
// Seção crítica
```

```
pthread_mutex_unlock(&lock);
```

Essa abordagem previne conflitos de acesso: O mutex garante que apenas uma thread acesse a um recurso de cada vez, evitando problemas de concorrência e dados inconsistentes.

4. Comandos e Interações

Comandos do Feed (Cliente)

Comando	Descrição
topics	Mostra os tópicos existentes, número de mensagens persistentes e estado.
msg <tópico> <duração> <mensagem>	Envia uma mensagem para o tópico especificado. A duração determina se a mensagem é persistente.
subscribe <tópico>	Subscrição de um tópico.
unsubscribe <tópico>	Cancelamento de subscrição de um tópico.
exit	Sai do programa, informando o manager.

Comandos do Manager (Administrador)

Comando	Descrição
users	Mostra utilizadores ativos.
remove <username>	Remove um utilizador da plataforma.
topics	Mostra os tópicos existentes e mensagens persistentes.
show <tópico>	Mostra todas as mensagens persistentes de um tópico.
lock <tópico>	Bloqueia o envio de mensagens para o tópico.
unlock <tópico>	Desbloqueia um tópico.
close	Encerra a plataforma, notificando os utilizadores.

5. Resultados e Validação

Casos de Teste

- **Cenário 1:** Criação e subscrição de tópicos.

```
bruno@LENOVO-BRUNO:~/TP-SO$ ./manager
Manager iniciado...
Admin> Utilizador pedro adicionado com sucesso.
Utilizador pedro subscreveu ao tópico futebol
[]

bruno@LENOVO-BRUNO:~/TP-SO$ ./feed pedro
Bem-vindo, pedro!
Ligação estabelecida com sucesso.
Comando> subscribe futebol
Subscrição ao tópico futebol efetuada com sucesso.
Comando> []
```

- **Cenário 2:** Envio de mensagens não-persistentes e persistentes.

```
bruno@LENOVO-BRUNO:~/TP-SO$ ./manager
Manager iniciado...
Admin> Utilizador pedro adicionado com sucesso.
Utilizador pedro subscreveu ao tópico futebol
Mensagem expirada removida do tópico escola
Mensagem expirada removida do tópico escola1
Utilizador maria adicionado com sucesso.
Distribuindo mensagem do tópico futebol para os subscritores.
Mensagem persistente guardada no tópico futebol
Mensagem enviada para pedro: Golo invrivel no último jog
o!

bruno@LENOVO-BRUNO:~/TP-SO$ ./feed pedro
Bem-vindo, pedro!
Ligação estabelecida com sucesso.
Comando> subscribe futebol
Subscrição ao tópico futebol efetuada com sucesso.
Comando> [Nova mensagem] Tópico: futebol, Autor: maria
Mensagem: Golo invrivel no último jogo!
Comando> []

bruno@LENOVO-BRUNO:~/TP-SO$ ./feed maria
Bem-vindo, maria!
Ligação estabelecida com sucesso.
Comando> msg futebol 120 Golo invrivel no último jogo!
120
Mensagem enviada para validação pelo Manager.
Comando> []
```

- **Cenário 3:** Bloqueio e desbloqueio de tópicos.

```
bruno@LENOVO-BRUNO:~/TP-SO$ ./manager
Manager iniciado...
Admin> Utilizador pedro adicionado com sucesso.
Utilizador pedro subscreveu ao tópico futebol
Mensagem expirada removida do tópico escola
Mensagem expirada removida do tópico escola1
Utilizador maria adicionado com sucesso.
Distribuindo mensagem do tópico futebol para os subscritores.
Mensagem persistente guardada no tópico futebol
Mensagem enviada para pedro: Golo invrivel no último jog
o!
Mensagem expirada removida do tópico futebol
lock futebol
Tópico futebol bloqueado por admin.
Admin> Distribuindo mensagem do tópico futebol para os s
ubscritores.
O tópico futebol está bloqueado, Mensagem não enviada.
[]

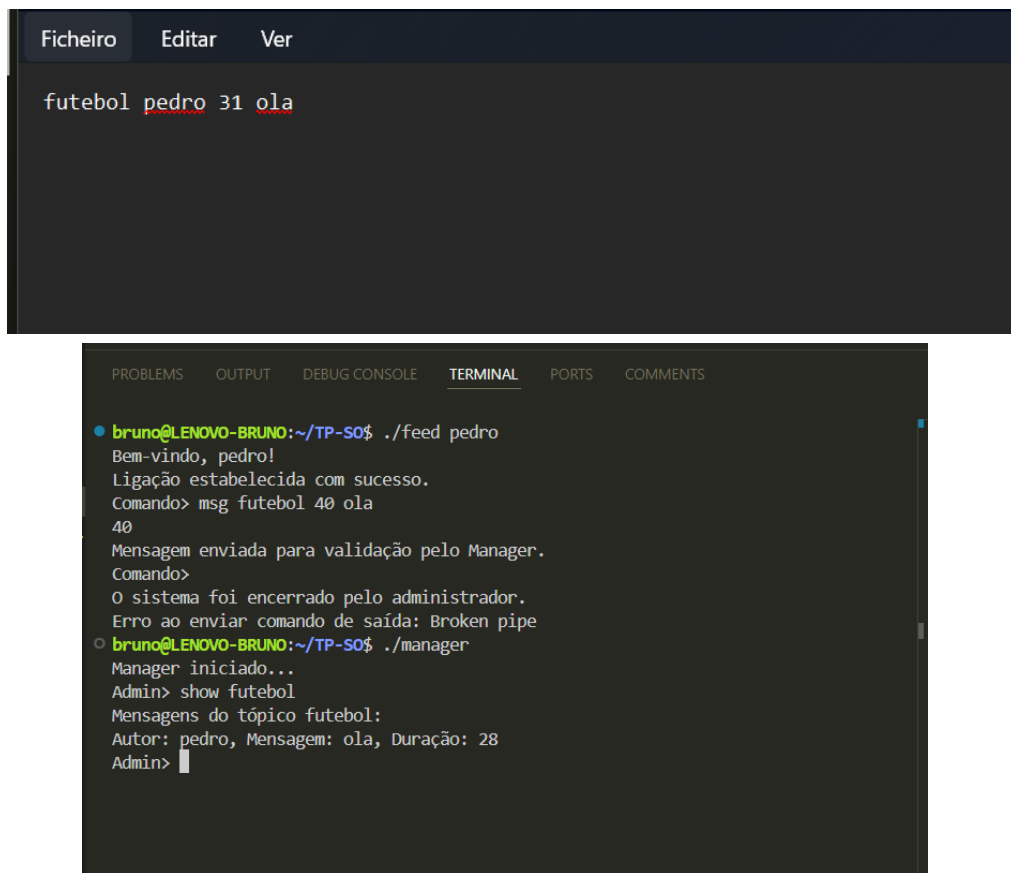
bruno@LENOVO-BRUNO:~/TP-SO$ ./feed pedro
Bem-vindo, pedro!
Ligação estabelecida com sucesso.
Comando> subscribe futebol
Subscrição ao tópico futebol efetuada com sucesso.
Comando> [Nova mensagem] Tópico: futebol, Autor: maria
Mensagem: Golo invrivel no último jogo!
Comando> O tópico futebol foi bloqueado
Comando> []

bruno@LENOVO-BRUNO:~/TP-SO$ ./feed maria
Bem-vindo, maria!
Ligação estabelecida com sucesso.
Comando> msg futebol 120 Golo invrivel no último jogo!
120
Mensagem enviada para validação pelo Manager.
Comando> msg futebol 0 ola
0
Mensagem enviada para validação pelo Manager.
Comando> [Erro] O tópico futebol está bloqueado. A mensagem não
foi enviada.
Comando> []
```

- **Cenário 4:** Recuperação de mensagens persistentes após reinício do manager.

```
bruno@LENOVO-BRUNO:~/TP-SO$ ./feed pedro
Bem-vindo, pedro!
Ligação estabelecida com sucesso.
Comando> msg futebol 40 ola
40
Mensagem enviada para validação pelo Manager.
Comando> O sistema foi encerrado pelo administrador.
Erro ao enviar comando de saída: Broken pipe
bruno@LENOVO-BRUNO:~/TP-SO$ []

bruno@LENOVO-BRUNO:~/TP-SO$ ./manager
Manager iniciado...
Admin> Utilizador pedro adicionado com sucesso.
Distribuindo mensagem do tópico futebol para os subscritores.
Mensagem persistente guardada no tópico futebol
close
Sistema encerrado.
bruno@LENOVO-BRUNO:~/TP-SO$ []
```



The top screenshot shows a graphical interface with a menu bar containing 'Ficheiro', 'Editar', and 'Ver'. Below the menu, the text 'futebol pedro 31 ola' is displayed, with 'pedro' and 'ola' underlined in red.

The bottom screenshot shows a terminal window with the following output:

```
bruno@LENOVO-BRUNO:~/TP-SO$ ./feed pedro
Bem-vindo, pedro!
Ligação estabelecida com sucesso.
Comando> msg futebol 40 ola
40
Mensagem enviada para validação pelo Manager.
Comando>
O sistema foi encerrado pelo administrador.
Erro ao enviar comando de saída: Broken pipe
bruno@LENOVO-BRUNO:~/TP-SO$ ./manager
Manager iniciado...
Admin> show futebol
Mensagens do tópico futebol:
Autor: pedro, Mensagem: ola, Duração: 28
Admin>
```

Exemplo de Interação

1. Utilizador pedro inicia o feed e subscreve o tópico futebol.
2. Utilizador maria envia a mensagem persistente:
msg futebol 120 Golo incrível no último jogo!
3. pedro visualiza imediatamente a mensagem no seu terminal.

6. Conclusão

O desenvolvimento deste projeto proporcionou uma compreensão aprofundada dos conceitos fundamentais de sistemas operativos, com foco na comunicação entre processos, sincronização e gestão de recursos. Durante a implementação, foram aplicadas diversas técnicas para assegurar o funcionamento correto do sistema, garantindo simultaneidade, consistência e eficiência.

Os principais aprendizados incluem:

- **Gestão de comunicação entre processos:** O uso de named pipes revelou-se eficiente para a troca de mensagens entre o feed e o manager, destacando a importância de uma arquitetura bem planeada para evitar bloqueios ou perda de dados.
- **Sincronização e paralelismo:** A utilização de mutexes e threads foi essencial para lidar com múltiplos utilizadores simultâneos, mostrando a importância de evitar condições de corrida.

- **Validação e controlo de limites:** A lógica de validação no manager para controlar o número máximo de tópicos, utilizadores e mensagens persistentes foi crucial para assegurar a estabilidade e escalabilidade do sistema.
- **Armazenamento persistente:** O mecanismo para guardar e carregar mensagens persistentes demonstrou a necessidade de garantir a consistência dos dados entre sessões do programa. Em resumo, o projeto não só consolidou os conhecimentos teóricos adquiridos, mas também demonstrou a relevância de soluções práticas e bem estruturadas em cenários reais. O resultado final é um sistema funcional, robusto e que reflete as melhores práticas no desenvolvimento de software para sistemas operativos.