

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Licenciatura em Engenharia Informática

TRABALHO PRÁTICO DE SISTEMAS OPERATIVOS

Simulação de Plataforma de Gestão de Táxis Autónomos

Autores:

Diogo Ribeiro Costa (Nº 2024143983)
Rodrigo Cravo Pereira (Nº 2024117439)

Ano Letivo 2025/2026

Índice	2
1. Introdução	3
2. Arquitetura do Sistema	3
2.1. Mecanismos de Comunicação (IPC)	3
3. Detalhes de Implementação e Fundamentação	4
3.1. Controlador: Concorrência e Sincronização	4
3.2. Veículo: Redirecionamento e Processos	4
3.3. Cliente: Multiplexagem de E/S	4
4. Tabela de Funcionalidades	4
5. Conclusão	4

1. Introdução

Este projeto visa a implementação de uma plataforma de gestão de transportes com veículos autónomos, baseada na arquitetura e primitivas do sistema operativo UNIX. O sistema simula o funcionamento de uma frota de táxis, clientes e um controlador central.

O desenvolvimento focou-se na aplicação prática dos conceitos teóricos e práticos lecionados, nomeadamente: gestão de processos, comunicação entre processos (IPC) via pipes, multiplexagem de E/S e sincronização de threads com mutexes.

2. Arquitetura do Sistema

A solução implementa uma arquitetura distribuída e modular, composta por três processos distintos:

- **Controlador (Servidor):** Entidade central responsável pela orquestração do sistema. Gere a estrutura de dados partilhada (viagens, utilizadores), controla a criação de processos filho (veículos) e gere o tempo simulado.
- **Cliente (Interface):** Processo que atua como interface com o utilizador. Implementa um modelo de E/S não bloqueante para enviar comandos e receber notificações assíncronas.
- **Veículo (Simulação):** Processo filho gerado dinamicamente pelo controlador. Opera em modo "Auto-Start", simulando o deslocamento e comunicando o seu estado através de canais de IPC.

2.1. Mecanismos de Comunicação (IPC)

A comunicação entre processos baseia-se estritamente nas primitivas UNIX estudadas:

Fluxo	Mecanismo	Justificação Teórica
Cliente -> Controlador	Named Pipe (FIFO Público)	Modelo Cliente-Servidor (N-para-1)
Controlador -> Cliente	Named Pipe (FIFO Privado)	Canal dedicado para resposta
Controlador -> Veículo	Argumentos (exec)	Passagem de parâmetros na criação do processo
Veículo -> Controlador	Pipe Anónimo + Redirecionamento	Comunicação Pai-Filho via stdout (dup2)
Veículo -> Cliente	Named Pipe do Cliente	Notificação direta de eventos
Sinais	SIGUSR1 / SIGINT	Controlo de fluxo assíncrono e terminação

3. Detalhes de Implementação e Fundamentação

3.1. Controlador: Concorrência e Sincronização

O Controlador adota uma arquitetura Multithreaded para lidar com múltiplas fontes de eventos simultâneas (Timer, Admin, Clientes), conforme o modelo de "Input com Threads" abordado nas aulas.

Dada a existência de dados partilhados (lista de viagens, contadores, estrutura DadosControl), foi identificada a existência de Secções Críticas. Para garantir a coerência dos dados e prevenir Race Conditions (Condições de Corrida), utilizou-se um Mutex Global (pthread_mutex_t trinco). Todas as operações de escrita ou leitura na estrutura partilhada são atómicas, estando protegidas por lock/unlock do mutex.

3.2. Veículo: Redirecionamento e Processos

A criação do veículo utiliza a sequência fork() seguida de exec(). Antes da substituição da imagem do processo, o descriptor STDOUT_FILENO é redirecionado para um pipe anónimo utilizando a primitiva dup2(). Isto permite que o Controlador leia a saída padrão do Veículo como se fosse um ficheiro, abstraindo a comunicação.

3.3. Cliente: Multiplexagem de E/S

Para evitar o bloqueio do cliente numa operação de leitura (seja no teclado ou no pipe), implementou-se o mecanismo de Multiplexagem de E/S através da system call select(). O select() monitoriza simultaneamente o descriptor STDIN e o descriptor do FIFO privado, desbloqueando o processo apenas quando um deles tem dados prontos a ler. Isto garante uma interface reativa e não bloqueante.

4. Tabela de Funcionalidades

Módulo	Funcionalidade	Estado
Geral	Arquitetura de 3 Processos	Cumprido
Controlador	Gestão de Utilizadores e Frota	Cumprido
Controlador	Sincronização com Mutexes	Cumprido
Veículo	Auto-Start e Redirecionamento	Cumprido
Cliente	Multiplexagem com select()	Cumprido
Comunicação	Named Pipes e Sinais	Cumprido

5. Conclusão

O trabalho foi concluído com sucesso, demonstrando a aplicação prática dos conceitos fundamentais de Sistemas Operativos. A solução desenvolvida gere eficazmente a concorrência através de threads e mutexes, garantindo a integridade dos dados partilhados.

A utilização correta de primitivas de IPC (Pipes e Sinais) e de mecanismos de E/S avançados (select, dup2) resultou num sistema robusto, capaz de simular um ambiente operacional complexo sem deadlocks ou bloqueios indevidos.