UNIVERSIDADE D
COIMBRA

Sistemas Operativos: Relatório

Rui Oliveira & Álvaro Terroso

Licenciatura em Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra

May 14, 2024

1 Introduction

Este relatório tem como intuito descrever os mecanismos de sincronização bem como armazenamento e funcionalidades de outros procedimentos. Nomeadamente, semáforos, organização de unnamed pipes e threads utilizadas . Cada membro utilizou 50 horas de trabalho.

2 Sistemas Utilizados

2.1 BackOffice

No BackOfficeUser apresentamos 2 processos distintos: o primeiro tem a função de receber inputs do terminal através de um ciclo while com scanf(); já o segundo processo tem a responsabilidade de ler os alertas ou estatísticas da Message Queue. Além disso, apenas é utilizado um signal handler para lidar com CTRL+C, garantindo o fecho de todos os processos e ponteiro do BACK_PIPE.

2.2 MobileUser

No Mobile User temos 2 processos distintos. O primeiro possui 3 threads diferentes, uma para cada tipo de mensagem a ser enviada para o named pipe USER_PIPE. Já o segundo é focado para ler as mensagens da Message Queue. Possui também um mutex que controla a quantidade de pedidos restantes a enviar.

2.3 SystemManager

2.3.1 Semáforos

sem_shared - controla o acesso ao array de users da estrutura da Shared Memory.
sem_userscount - controla o acesso à contabilização de users totais ativos.
sem_read_count. - controla o array responsáel por gerir a disponibilidade dos unnamed pipes.
sem_plafond - controla o acesso ao plafond de cada user dentro do arrya de users.
log_mutex - controla o acesso e a escrita no ficheiro log.
sem_statics - controla o acesso às estatisticas gerais de consumo dos users na Shared Memory.
sem_monitor - controla e gere quando o processo Monitor Engine tem acesso aos valores do plafond da Shared Memory.
sem_flag - controla o valor da variável responsável por avisar o monitor engine que a lista de utilizadores está cheia.
sem_run - controla o acesso à variável *run*, vital para todo o programa correr.
sem_times - controlia o acesso à variável *times*, que conta o número de vezes que o mobile user recebe alertas de plafond.
sem_go - avisar à thread sender que a thread receiver adicionou pedidos às filas

2.3.2 Pthread Mutexes

mut_video - este mutex controla o acesso à fila de video.
mut_other - este mutex controla o acesso à fila de other.

2.3.3 Shared Memory

A estrutura da Shared Memory é composta por uma estrutura de estatísticas, que envia de 30 em 30 segundos e por pedido estatísticas ao BackOfficeUser. Também por outra estrutura de users_ que armazena todos os MobileUsers loggados e variáveis de controlo, como um boolean run, um int flag para saber quando o mobile user não conseguiu dar login pela fila estar cheia, um int mobile_users que possui o número de mobile users registados, um int adicional que representa o unnamed pipe extra quando as filas estão cheias e um array read_count_shared, que é um array de unnamed pipes disponiveis.

2.3.4 Armazenamento

Utilizamos um array de inteiros inicializados a 0, que estão diretamente ligados com os unnamed pipes. Quando um pipes possui uma mensagem, a sua posição no array fica a 1, até ficar livre, onde retorna para 0. As filas de mensagens são representadas também por uma struct, que possui a mensagem a ser passada, do estilo FIFO.

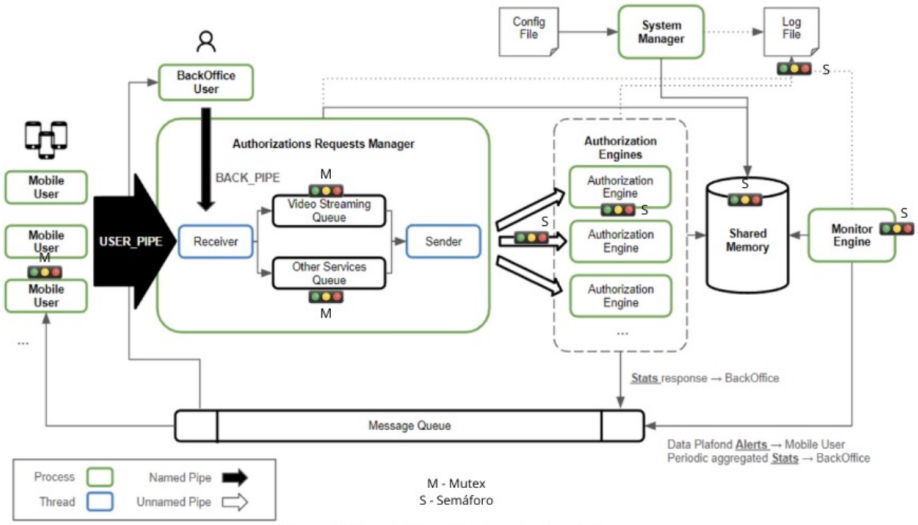

Figura 2: Arquitetura técnica do simulador