



RELATÓRIO PROJETO

Sistemas Operativos

Síntese

Neste projeto devemos simular um sistema de autorização em tempo real onde vários utilizadores móveis tentam utilizar vários serviços de dados em simultâneo


Henrique Oliveira

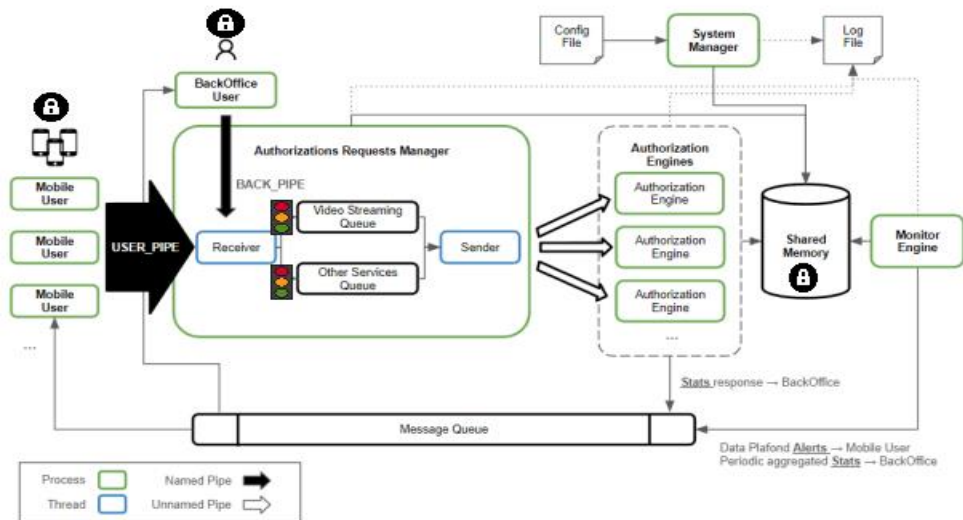
2022211169

Luis Vaz

2022214707

Esquema da Arquitetura:

 = mutex



- O uso de um mutex na shared memory tem como objetivo coordenar o acesso concorrente de processos a essa região da memória, garantindo que apenas um processo de cada vez esteja realizando operações críticas nessa área. Isso é essencial para garantir que não há inconsistências nos dados, uma vez que vários processos podem tentar ler ou escrever na memória compartilhada simultaneamente.
- Para o controlo do acesso à Video Streaming Queue e à Other Services Queue, implementou-se um semáforo binário dentro da estrutura que define a fila. Desta forma, garantimos que apenas um processo ou thread consegue alterar a respetiva fila de cada vez. A implementação do semáforo é feita com `sem_init` para inicialização e `sem_wait` e `sem_post` para controle do acesso.
Além disso, utilizamos uma variável condicional `pthread_cond_t` `queues_not_empty` e um mutex `pthread_mutex_t` `mutex` para gerir o estado da fila, ou seja, se está vazia ou cheia. Cada vez que um pedido é inserido, sinalizamos a variável condicional para informar que a fila não está vazia. Na função `enqueue`, depois de adicionar um pedido à fila, a variável condicional é sinalizada dentro de um bloqueio de mutex. Na função `dequeue`, utilizamos `sem_wait` para bloquear a fila durante a remoção de um pedido, garantindo que a remoção só acontece se a fila não estiver vazia.

- No contexto do problema, o Mobile User precisa gerir a geração e envio de múltiplos pedidos de autorização de serviço em intervalos periódicos. Isso é realizado por meio de threads distintas, cada uma responsável por um tipo de serviço (vídeo, música e social). Além disso, uma quarta thread fica encarregada de ouvir a fila de mensagens para receber alertas. A utilização do mutex torna-se essencial nesse cenário para garantir a sincronização e a consistência dos dados compartilhados entre essas threads concorrentes.
- No BackOffice existe uma thread é responsável por enviar comandos ao Authorization Request Manager, enquanto outra thread é responsável por ouvir a fila de mensagens e processar as estatísticas recebidas. O uso do mutex é crucial para garantir que essas operações concorrentes não interfiram umas com as outras, mantendo a consistência e a integridade dos dados compartilhados.
- Para evitar a espera ativa, onde a thread ficaria em um loop contínuo verificando se há novos pedidos nas filas, uma variável de condição (`queues_not_empty`) e um mutex (`mutex`) são utilizados. A `pthread_cond_wait` é usada para bloquear a thread até que seja sinalizada que há um novo pedido em uma das filas. O mutex é bloqueado antes de entrar na espera e desbloqueado após a thread ser acordada. Isso evita que a thread fique em um loop contínuo (espera ativa) consumindo CPU desnecessariamente.

Tempo despendido por elemento do grupo:

Luis Miguel Ferreira Vaz (2022214707) – 68 horas

Henrique de Araújo Oliveira(2022211169)- 64 horas