UNIVERSIDADE FEDERAL DE PELOTAS
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO
Sistemas Operacionais

Relatório Final

Felipe L. K. Santana, Lucas Superti da Silva, Mateus Henrique Schröpfer,
Jonatha Viegas da Silva

Pelotas
2025

1 Introdução

A sincronização de processos e threads é essencial na computação concorrente, pois garante que múltiplas tarefas possam acessar recursos compartilhados de forma organizada, evitando erros como condições de corrida.

Este relatório descreve a implementação de um sistema de simulação de alocação de recursos em uma lan house, utilizando threads e semáforos em linguagem C. O objetivo é modelar o comportamento de diferentes tipos de clientes (Gamers, Freelancers e Estudantes) que competem por recursos limitados (PCs, óculos de realidade virtual e cadeiras especiais), evitando condições de deadlock e garantindo justiça na alocação.

A simulação foi desenvolvida para:

- Controlar o acesso concorrente aos recursos.
- Evitar deadlocks por meio de estratégias de aquisição ordenada de recursos.
- Coletar estatísticas sobre utilização de recursos e tempo de espera dos clientes.

2 Desenvolvimento da Solução

2.1. Estrutura Geral do Sistema

O sistema consiste em:

- Recursos compartilhados:
  - NUM_PCS (10 computadores)
  - NUM_VR (6 óculos de realidade virtual)
  - NUM_CADEIRAS (8 cadeiras especiais)
- Tipos de clientes:
  - Gamer: Requer PC, VR e cadeira.
  - Freelancer: Requer PC e cadeira (opcionalmente VR).

- Estudante: Requer apenas PC.

2.2. Mecanismo de Controle de Concorrência

- Semáforos**:** Controlam o acesso aos recursos (semPcs, semVr, semCadeiras).
- Mutexes**:** Protegem variáveis compartilhadas (estatísticas e contadores de uso).
- Algoritmo de Aquisição de Recursos:
  - Cada cliente tenta adquirir os recursos em uma ordem específica para evitar deadlock.
  - Se um recurso não estiver disponível, o cliente libera os já adquiridos e tenta novamente após um tempo aleatório.

2.3. Teste Forçado de Deadlock

Antes de iniciar a simulação principal, o programa realiza um teste controlado de deadlock para verificar se o sistema consegue evitar impasses. Esse teste é implementado na função main() e segue os seguintes passos:

1. Alocação Artificial de Recursos:
   - Quase todos os PCs e VRs são reservados (NUM_PCS-1 e NUM_VR-1), deixando apenas um de cada disponível.
   - Isso cria uma situação crítica onde dois clientes (um Gamer e um Freelancer) competem pelos recursos restantes.
2. Início das Threads de Clientes:
   - Duas threads são criadas:
     - Gamer**:** Tenta adquirir PC → VR → Cadeira.
     - Freelancer**:** Tenta adquirir PC → Cadeira → VR.
   - Como os recursos estão quase esgotados, uma das threads pode ficar bloqueada temporariamente.
3. Resolução do Deadlock:
   - Após 5 segundos, os recursos reservados são liberados.
   - O sistema verifica se as threads conseguiram prosseguir sem travar permanentemente.

- Se o programa continuar executando, significa que o mecanismo de prevenção de deadlock funciona.

2.4. Threads de Clientes

Cada cliente é representado por uma thread que:

1. Tenta adquirir os recursos necessários.
2. Registra o tempo de espera.
3. Simula o uso dos recursos por um período aleatório (15 a 60 minutos).
4. Libera os recursos ao finalizar.

2.5. Coleta de Estatísticas

O sistema registra:

- Número de clientes atendidos e que falharam.
- Tempo médio de espera.
- Utilização total de cada recurso por tipo de cliente

3. Passos para Compilar

O código foi desenvolvido em C e utiliza a biblioteca pthread para threads e semáforos.

Compilação no Linux

bash

Copy

```
gcc -o main main.c -lpthread -lm
```

Execução

bash

Copy

```
./main
```

4. Dificuldades Encontradas

- Deadlock: Inicialmente, a ordem de aquisição de recursos poderia levar a impasses. Isso foi resolvido implementando uma estratégia de liberação segura quando um recurso não estava disponível.
- Precisão do Tempo: A conversão entre tempo real e tempo simulado exigiu ajustes para evitar inconsistências.
- Controle de Concorrência: Garantir que as estatísticas fossem atualizadas corretamente sem condições de corrida exigiu o uso de mutexes adicionais.

5. Conclusão e Sugestões para Trabalhos Futuros

5.1. Conclusão

O sistema simula eficientemente o gerenciamento de recursos em uma lan house, evitando deadlocks e coletando métricas relevantes.

Melhorias Possíveis:

- Priorização de Clientes: Implementar filas prioritárias para diferentes tipos de clientes.
- Escalabilidade: Testar o sistema com um número maior de recursos e clientes.
- Interface Gráfica: Desenvolver uma visualização em tempo real da simulação.
- Balanceamento Dinâmico: Ajustar automaticamente a quantidade de recursos com base na demanda.

Este projeto demonstra a aplicação de conceitos de sistemas concorrentes em um cenário prático, sendo uma base para estudos mais avançados em otimização de recursos computacionais.

5.2. Sugestão de Trabalho Futuro

Simulação de Tráfego em um Cruzamento

Objetivo: Controlar carros (threads) passando por um cruzamento com semáforos.

Recursos:

- Semáforos (verde/vermelho) para cada direção.
- Mutex para evitar colisões no centro do cruzamento.
  Desafios:
- Priorizar direções (ex.: via principal vs. secundária).
- Evitar deadlocks (ex.: todos os carros travados esperando).