PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

RAPHAELA DE SOUZA RIBEIRO

PROGRAMAÇÃO PARALELA E DISTRIBUÍDA: ATIVIDADE AVALIATIVA 02 - RELATÓRIO

CAMPINAS
2024

Relatório Final sobre Sincronização e Coordenação em Programação Paralela

Introdução

A programação paralela é uma abordagem que permite a execução simultânea de código, aumentando a eficiência em sistemas de multiprocessamento. Em ambientes onde múltiplas threads acessam recursos compartilhados, a sincronização é fundamental para garantir a consistência dos dados e evitar problemas como condições de corrida e deadlocks. Este relatório aborda os conceitos de *Leitores e Escritores*, Barreiras, Sinais e Deadlocks, e inclui exemplos práticos em C utilizando `pthread`.

Conceitos

1. **Leitores e Escritores** O problema de *Leitores e Escritores* visa gerenciar o acesso concorrente a um recurso compartilhado, permitindo que múltiplos leitores possam acessar simultaneamente enquanto um escritor requer acesso exclusivo. Este conceito é importante em sistemas de banco de dados e logs onde a consistência dos dados deve ser preservada sem comprometer a eficiência de leitura.
2. **Barreiras** Barreiras são pontos de sincronização nos quais um conjunto de threads deve parar e esperar até que todas as outras threads alcancem esse ponto antes de prosseguir. É útil em algoritmos paralelos que precisam ser sincronizados em fases específicas, como aplicações de processamento de dados e computação distribuída.
3. **Sinais** Sinais são usados para notificar threads de que uma condição foi atendida, permitindo a continuação da execução de outras *threads*. Este mecanismo é comumente usado para implementar filas de tarefas e sistemas de monitoração onde uma thread deve esperar até que uma condição específica seja satisfeita.
4. **Deadlocks** Um deadlock ocorre quando duas ou mais threads entram em um estado de espera mútua, onde nenhuma delas pode prosseguir. Situações de deadlock são comuns em sistemas que utilizam múltiplos bloqueios. A prevenção e detecção de deadlocks envolvem técnicas como ordenação de recursos e monitoramento de bloqueios.

Implementações Práticas

1. **Exemplo de Leitores e Escritores** O código em C implementado utiliza `pthread_mutex_t` e `sem_t` para gerenciar o acesso de threads de leitores e escritores a um recurso compartilhado. *Leitores* podem acessar o recurso simultaneamente, enquanto escritores bloqueiam totalmente o acesso.
2. **Exemplo de Barreiras** Implementação de um ponto de sincronização onde as threads esperam até que todas cheguem antes de continuar. Utilização de variáveis condicionais e mutexes para gerenciar a barreira.
3. **Exemplo de Sinais** O exemplo inclui o uso de `pthread_cond_wait()` e `pthread_cond_signal()` para criar uma comunicação entre threads, mostrando como uma thread pode ser colocada em espera até que outra envie um sinal para despertar.
4. **Exemplo de Deadlock e Prevenção** O código simula um deadlock e inclui uma solução que utiliza uma ordem de solicitação de recursos para evitar o problema.

Funcionamento dos Programas

- **Leitores e Escritores**: A primeira thread leitora bloqueia o acesso à escrita. Quando não há mais leitores, o recurso é liberado para escritores.
- **Barreiras**: Threads só prosseguem após todas terem alcançado o ponto de sincronização.
- **Sinais**: Threads esperam uma condição ser satisfeita para prosseguir.
- **Deadlock**: Demonstração de bloqueios mútuos e prevenção com ordenação de recursos.

Referências Bibliográficas

- Tanenbaum, A. S., & Bos, H. (2024). *Modern Operating Systems*.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts*.
- Butenhof, D. R. (1997). *Programming with POSIX Threads*.
- Raynal, M. (2013). *Concurrent Programming: Algorithms, Principles, and Foundations*.
- Coffman, E. G., Elphick, M. J., & Shoshani, A. (1971). "System Deadlocks." *ACM Computing Surveys*.
- Kerrisk, M. (2010). *The Linux Programming Interface*.

Referências de Bibliotecas Utilizadas

- `pthread.h`: Biblioteca para manipulação de threads POSIX.
- `semaphore.h`: Biblioteca para controle de semáforos.

Link do repositório no GitHub

- https://github.com/RaphaRibeiro0810/atv-aval-2-prog-paral-distrib.git