Universidade Presbiteriana Mackenzie

Faculdade de Computação e Informática

Sistemas Operacionais

Trabalho II: Threads

1. Introdução

As threads, também conhecidas por “processos leves”, são linhas de execução distintas em um mesmo processo. Todas as *threads* de um processo compartilham a mesma região de memória, apesar de possuírem alguns atributos individuais (como contador de programa e pilha). Isto permite que os dados sejam manipulados em paralelo, o que pode acelerar a execução do processo em ambientes multiprocessados, ou permitir a existência de várias instâncias de um mesmo processo (como *queries* em um banco de dados ou requisições a um servidor web). Justamente nesta capacidade reside o maior problema da execução concorrente: as condições de corrida. Este problema ocorre quando diversas *threads* manipulam o mesmo dado sem nenhuma restrição, deixando-o em um estado inconsistente.

Neste trabalho você irá desenvolver um programa com diversas threads e resolver as potenciais condições de corrida. O trabalho pode ser feito individualmente ou em dupla.

2. Metodologia

Em um canteiro de obra existem estoques de cimento e tijolos (para efeito de simplificação do modelo, consideremos que apenas estes dois são suficientes para construir casas). Neste terreno serão construídas C casas por P pedreiros. Cada casa necessita de C_c quilos de cimento e C_t tijolos para sua construção.

Os pedreiros seguem rigorosamente os seguintes passos:

1. Dirige-se ao estoque de cimento
2. Se algum outro trabalhador estiver manipulando o estoque, espera até que ele acabe
3. Busca P_c quilos de cimento no estoque de cimento e coloca em seu carrinho de mão em tempo P_{tc}
4. Dirige-se ao estoque de tijolos
5. Se algum outro trabalhador estiver manipulando o estoque, espera até que ele acabe
6. Busca P_t tijolos no estoque de tijolos e coloca em seu carrinho de mão em tempo P_{tt}
7. Escolhe uma das casas (escolha pode ser randômica)
8. Leva o carrinho de mão com o material para aquela casa
9. Se algum outro pedreiro já estiver naquela casa, espera até que ele termine seu serviço
10. Realiza o serviço necessário em tempo P_{ts}
11. Volta para o estoque de cimento com o carrinho de mão vazio para buscar mais cimento

Escreva um programa utilizando a biblioteca *pthreads.h* que modele a obra descrita acima. O programa deve ler do usuário os valores de C, P, C_c, C_t, P_c, P_t, P_{tc}, P_{tt} e P_{ts}. Para cada pedreiro, deve ser criada uma thread que realiza a rotina do pedreiro. Os estoques devem ser exatamente suficientes para a construção de todas as casas. O sincronismo entre os pedreiros deve ser implementado com semáforos binários (i.e. *mutex*). As variáveis P_{tx} representam um *delay* de tempo em segundos. Este *delay* pode ser implementado com uma chamada a função `sleep()`.

A saída de seu programa deve ser formatada em colunas, uma para cada pedreiro. A cada novo passo executado da rotina do pedreiro, deve ser mostrada na saída uma mensagem descrevendo a ação, em sua respectiva coluna. Abaixo, um possível exemplo de saída:

Tabela 1: Exemplo de saída para 4 pedreiros

Pedreiro 1	Pedreiro 2	Pedreiro 3	Pedreiro 4
Carregando cimento	Esperando cimento	Esperando cimento	Esperando cimento
Est. cimento: 1000			
Carregando tijolos	Carregando cimento		
Est. tijolos: 1000	Est. cimento: 900		
Construindo casa 1	Carregando tijolos	Carregando cimento	
Cimento casa 1: 100	Est. tijolos: 900	Est. cimento: 800	
Tijolos casa 1: 100	Construindo casa 2	Carregando tijolos	Carregando cimento
	Cimento casa 2: 100	Est. tijolos: 800	Est. cimento: 700
Carregando cimento	Tijolos casa 2: 100	Construindo casa 1	Carregando tijolos
Est. cimento: 600		Cimento casa 1: 200	Est. tijolos: 700

Junto do código deve ser entregue um arquivo texto contendo sua documentação. Este arquivo deve possuir, essencialmente, uma descrição em português do uso das *threads* e dos semáforos (quais são e o propósito de cada um).

3. Entrega

A entrega será feita pelo Moodle, até o dia estipulado no link disponível para envio. Deverá ser enviado um arquivo com o nome:

SO_T1_nomealuno1_nomealuno2_nomealuno3.zip

contendo:

1. O(s) arquivo(s)-fonte, com comentários
2. Um MakeFile para compilação
3. Documentação do código (em formato .doc/.docx ou .pdf)

4. Pontuação

A pontuação será de acordo com os critérios abaixo. O compartilhamento de código-fonte entre diferentes trabalhos (total ou parcial) implica em uma nota **zero**. Trabalho entregues com atraso não serão aceitos.

- Funcionalidade: 80%
- Documentação: 20%