Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação

Threads
Simulação de Cache
(Tarefa para Unidade I)

Disciplina: DCA0108 – Sistemas Operacionais
Turma: 01, 2022.2
Horário: 24M12
Professor: Diogo Pinheiro Fernandes Pedrosa (diogo.pedrosa@ufrn.br)

Contextualização e Problema Abordado

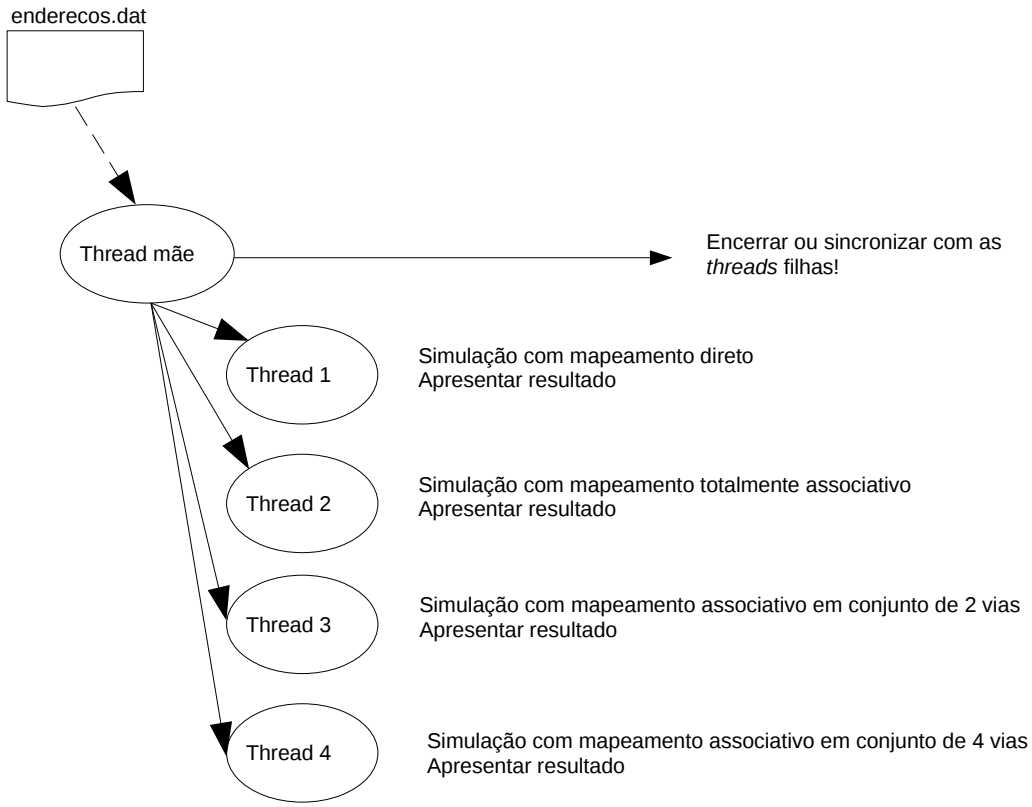
Thread, ou programação com *thread*, consiste em uma forma de fazer com que tarefas que pertençam a um mesmo processo possam ser executadas de forma independente umas das outras. Programas *multithread* possuem vários fluxos ou linhas de execução e, em geral, apresentam um conjunto de vantagens sobre programação com processos via comunicação interprocessos.

A manipulação das *threads* é feita pelo programador através de diversas funções que são definidas por uma biblioteca específica para a linguagem escolhida. O domínio de programação com *threads* permite com que as aplicações desenvolvidas sejam eficientes quando executadas em um computador.

Esta atividade consiste na elaboração de um programa *multithread* para realizar a simulação de mapeamento de uma cache simples (única), considerando a diversidade existente de funções de mapeamento que possam ser utilizadas. A figura a seguir sintetiza o trabalho a ser desenvolvido.

O programa inicial (que será a *thread* mãe) deverá realizar a leitura de um arquivo texto previamente fornecido (nomeado `enderecos.dat`, presente na turma virtual no SIGAA) e salvar suas informações em uma estrutura de dados adequada para o problema. Os dados lidos são, para efeitos de simulação de cache, uma listagem de endereços de uma memória de 16 K posições, os quais são acessados para operações de leitura. Estando, então, a *thread mãe*, com a estrutura de dados que representa os endereços, ela deve lançar quatro *threads* em que, cada uma, realizará uma simulação de cache diferente. As simulações e especificações são:

- A primeira *thread* deve simular os acertos e falhas de uma cache com mapeamento direto. Essa cache deve ter 1 K linhas, com 4 endereços por linha (ou seja, os blocos da memória principal têm 4 endereços de tamanho);
- A segunda *thread* deve simular acertos e falhas de uma cache com mapeamento totalmente associativo. Esta cache também tem 1 K linhas de tamanho, com 4 endereços por linha;
- A terceira *thread* deve simular os acertos e falhas em uma cache com mapeamento associativo em conjunto de 2 vias. Neste caso, a cache deve ser estruturada em 512 conjuntos. Cada via, de cada conjunto, tem 4 endereços de espaço; e
- A quarta e última *thread* deve simular acertos e falhas em uma cache com mapeamento associativo em conjunto também, mas com 4 vias. Dessa forma, a cache vai apresentar um total de 256 conjuntos. Cada via dos conjuntos tem 4 endereços de espaço.



A execução das *threads* deve, basicamente, realizar a contagem de acertos e falhas e, no final, antes de seu encerramento, exibir em tela as estatísticas obtidas (percentual de acertos e também de falhas). Neste caso, pode-se optar para que a *thread* mãe encerre (finalize) após criar as quatro *threads* filhas. Neste caso, deve-se observar se a função de finalização da *thread* mãe não vai implicar a finalização de todas as filhas criadas.

Uma outra opção seria sincronizar a *thread* mãe com a execução das filhas (colocando-a em espera por meio de uma função *join*) e, no final, fazer com que a *thread* mãe seja a responsável por apresentar os resultados das quatro simulações realizadas (as *threads* não exibiriam o resultado, mas o disponibilizaria em uma variável global para que pudesse ser usada pela *thread* mãe).

O objeto de avaliação para atribuição da nota da unidade será um relatório em que devem estar apresentadas:

1. Uma introdução fazendo a apresentação do problema;
2. Uma seção explicando a biblioteca e as funções para gerenciamento de *threads* que foi escolhida;
3. Uma seção explicando a construção do código do programa, apresentando as dificuldades encontradas (se pertinente) e as soluções utilizadas;
4. Uma seção final com a apresentação dos resultados e análise sobre a programação *multithread* com a biblioteca escolhida; e
5. Referências consultadas.

Percebam que a teoria relativa à memória cache não é objeto de avaliação neste trabalho. Então, não é necessário aprofundar-se na teoria deste tema.

Esta é uma tarefa em grupo, com equipes até 3 componentes. Caso exista a necessidade de grupos maiores, o trabalho desenvolvido deverá aumentar a quantidade e complexidade de simulações realizadas, testando, assim, novas configurações de cache (caches maiores, menores, com 2 níveis, com mais vias, etc.). Não é necessário enviar o código desenvolvido como arquivo separado para execução. Ele pode ser inserido como anexo do relatório (fazendo parte do texto do relatório) ou por meio de *link* para repositório de códigos-fontes.

Devido à facilidade existente de simular caches com a *engine pyCacheSim*, recomenda-se elaborar o código com a linguagem *Python*. No entanto, o grupo pode escolher outras linguagens se assim desejar.

O envio do relatório será feito pelo SIGAA, na parte de Tarefas, até a data limite especificada.

Em caso de dúvidas, enviar mensagem para diogo.pedrosa@ufrn.br.