# Atividade Prática - OpenMP

## Usando OpenMP

Para que seja possível usar os **#pragma**s do OpenMP é necessário usar uma flag especial de compilação: -fopenmp. No Makefile essa flag é setada na variável CFLAGS:

```
CFLAGS=-fopenmp
```

Para usar as funções do runtime OpenMP (aquelas que começam com **omp**_) será necessário um #include:

```
#include <omp.h>
```

O número de threads a ser usado por um programa OpenMP é determinado de várias formas:

1. Variável de ambiente **`OMP_NUM_THREADS`**
2. Chamada da função **`omp_set_num_threads(`**`novo_valor`**`)`**
3. Cláusula **`num_threads`**`(valor)`, usada em um `#pragma omp paralell`

As alternativas subsequentes sobrescrevem as alternativas listadas antes. Se `OPENMP_NUM_THREADS=`**`8`**, e a função `omp_set_num_threads(`**`16`**`)` é chamada, serão usadas **16** threads.

## Exercício 1

Um engenheiro escreveu o programa presente no arquivo `main.c`. Só o engenheiro sabe o que esse código faz, mas você pode observar que cada iteração do `for` é independente de todas as demais. Utilize OpenMP para paralelizar esse programa.

O programa lê o número de threads e o tamanho do vetor a ser calculado como argumentos de linha de comando. A implementação em OpenMP deve usar o número de threads indicado na variável **`n_threads`**. O programa também inclui uma medida do tempo gasto realizando o cálculo. Compute o *speedup* do seu programa executando-o com diferentes números de threads.

## Exercício 2

O programa fornecido inicializa duas matrizes quadradas com o tamanho fornecido como argumento de linha de comando e as multiplica. Esse programa foi paralelizado com OpenMP por um estagiário. O cliente ligou enfurecido no suporte dizendo que o programa não está tão rápido quanto deveria, e que ele produz resultados incorretos. Arrume o(s) erro(s) cometido(s) pelo estagiário de modo que o cliente fique satisfeito.

**Atenção**: apenas a paralelização da função **`mult_matrix`** será avaliada. A função `main()` não será executada pelo script corretor. A função **`mult_matrix`** será chamada diretamente

Uma vez resolvido o problema, **explique pro estagiário o que ele fez de errado** (texto fornecido direto no Moodle, durante a submissão). Seja didático, para que o estagiário compreenda o problema e não volte a cometer o mesmo erro...

## Exercício 3

O programa fornecido calcula o desvio padrão de uma sequência de números gerados aleatoriamente. O tamanho da sequencia é lido como argumento da linha de comando. Paralelize a computação realizada pela função **`standard_deviation()`** usando OpenMP.

Bônus: paralelize a geração de números aleatórios, de modo a evitar que as threads gerem números repetidos. Use a função `rand_r(unsigned*)` e gere um *seed* para cada thread.

## Correção automática

O script de correção (**grade-openmp.sh**) está dentro do esqueleto e deve ser executado na própria pasta onde está. Dentro da pasta de cada exercício há um script específico (**grade-openmp-ex-*.sh**) daquele exercício. Leia os enunciados com atenção e modifique as funções dadas. **Não as renomeie**, ou o script corretor acusará erro de compilação.

Se houver influência do ambiente, o script pode erroneamente abaixar ou aumentar a nota. Se o script só atribuir 10 esporadicamente, certamente a solução não está 100% correta.

# Entrega do Exercício

Submeta um arquivo **.tar.gz (ou zip)** na **mesma estrutura do esqueleto**dado como ponto inicial nessa tarefa. O uso do esqueleto fornecido é **obrigatório**. Em especial, você deve criar suas soluções editando os arquivos dentro das pastas **exercicio_*** . Utilize **make submission** para garantir que será gerado um arquivo conforme solicitado.

O prazo para entrega é **14 de Outubro às 23h55**.