

Programação Concorrente

2024/2025

Projeto – Parte B

O processamento computacional de grandes conjuntos de dados é normalmente composto por um conjunto de tarefas de elevada complexidade temporal. No entanto, muitas vezes pode-se tirar partido da existência de múltiplos CPUs ou *Cores* para reduzir o tempo total de processamento.

Um desses exemplos é a conversão e processamento de imagens. Normalmente em bancos de imagens, todas as imagens sofrem um conjunto de conversões igual: criação de uma cópia com menor resolução, criação de um *thumbnail*, ou mesmo aplicação de filtros, por exemplo. Estes passos são aplicados de forma igual a todas as imagens ou fotografias.

Na primeira parte do Projeto os alunos atribuíram a cada *thread* um conjunto pré-determinado de imagens. Nesta parte os alunos utilizarão um mecanismo dinâmico em que as *threads*, quando desejam iniciar o processamento de uma nova imagem, vão buscá-la à lista de imagens.

Se na parte A cada *thread* recebia sempre as “mesmas imagens”, dependendo apenas da ordem e posição na lista, nesta parte *threads* que processem imagens mais pequenas, processarão mais imagens: mal termina o processamento de uma imagem, essa *thread* irá à lista receber a próxima imagem a ser processada.

1 Descrição da parte B do projeto

Neste projeto os alunos deverão alterar a aplicação **old-photo-parallel-A** de modo a usar um mecanismo dinâmico de atribuição de imagens a cada *thread*.

A aplicação resultante deverá efetuar as 4 transformações a uma série de imagens armazenadas numa diretoria para produzir imagens com efeito antigo. Esta aplicação chamar-se-á **old-photo-parallel-B**.

De modo a acelerar o processo e reduzir o tempo de processamento deverão ser usadas *threads*.

1.1 Funcionamento geral

Ao executar a aplicação desenvolvida, o utilizador indica na linha de comandos o nome da pasta onde se encontram as imagens a serem processadas. A aplicação, depois de saber que imagens se encontram na pasta indicada, seleciona as terminadas em **.jpeg** e, para cada uma destas imagens, produz uma imagem com efeito antigo. As imagens resultantes terão o nome da imagem original, mas serão armazenadas em diretorias específicas tal como descrito na Secção 2.3

1.2 Paralelização

A aplicação paralela a desenvolver (chamada **old-photo-parallel-B**), deverá utilizar um determinado número de *threads* trabalhadoras para transformar as imagens.

Ao contrário do **old-photo-parallel-A**, no **old-photo-parallel-B** as *threads* trabalhadoras não sabem, *a priori* quais as imagens a si atribuídas. Neste projeto estas *threads* implementam o seguinte pseudocódigo

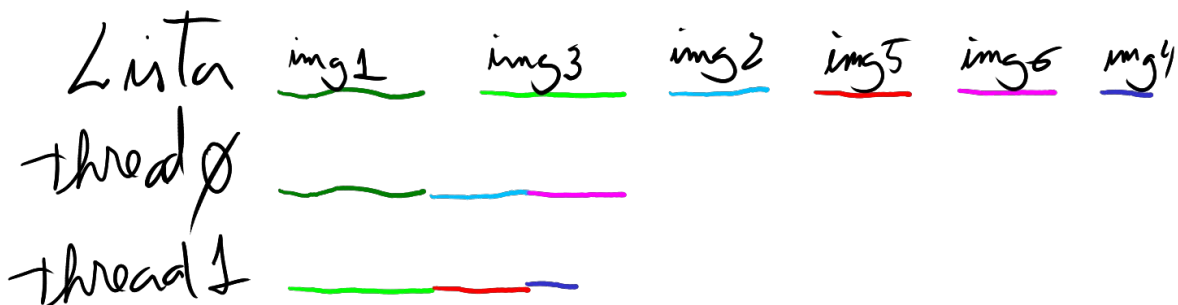
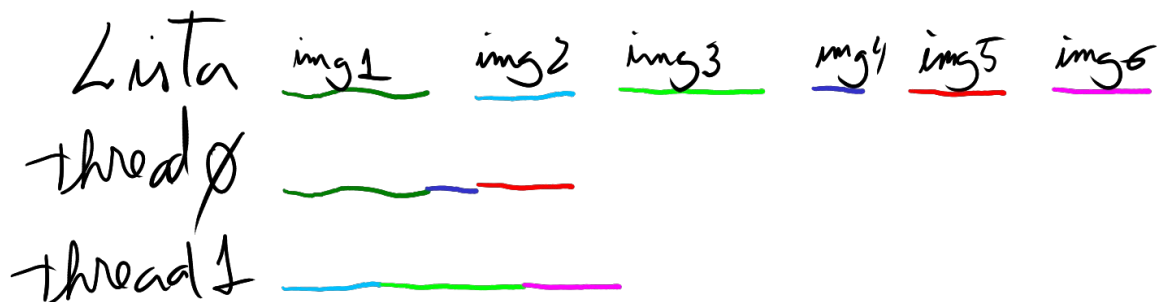
```
while (existem imagens a ser processadas){
    seleciona próxima imagem a ser processadas
    transforma imagem
}
```

A seleção da próxima imagem ser processada por cada *thread*, poderá ser implementada através da leitura do identificador da próxima imagens a partir de um **pipe**, ou através de um contador global.

Cada *thread* trabalhadora irá processa um subconjunto disjunto de imagens e para cada uma das imagens por si processadas gera a imagem envelhecida (exatamente como no **old-photo-serial.c**). O número de *threads* criadas é definido pelo utilizador através de um argumento da linha de comandos. O utilizador também define por que ordem as imagens são guardadas inicialmente na memória (alfabeticamente ou por tamanho crescente).

Depois de saber os nomes das imagens que se encontram na pasta e ordená-las (por tamanho ou alfabeticamente), o main cria um conjunto de *threads* trabalhadoras.

A ordem de início de processamento das imagens corresponderá à ordem pela qual se encontram armazenadas na lista, mas a atribuição das imagens às *threads* dependerá da disponibilidade de cada *thread*, como se exemplifica nos seguintes exemplos em que 6 imagens (de tamanhos e tempos de processamentos diferentes) serão processadas por 2 *threads*:



Quando não houver mais imagens para serem processadas as *threads* trabalhadoras criadas anteriormente deverão terminar.

O *main* deverá esperar pela terminação de todas as *threads* antes de sair.

Durante a execução da aplicação **old-photo-parallel-B**, o utilizador poderá carregar na tecla **S** para ver no ecrã quantas imagens já foram transformadas (como descrito na Secção 2.2).

2 Funcionamento da aplicação paralela

A aplicação deverá ser desenvolvida em **C** e executará em Linux, WSL ou MAC OS X.

2.1 Argumentos da linha de comandos

Para a execução da aplicação **old-photo-parallel-B** o utilizador deverá sempre indicar através dos argumentos da linha de comandos o seguinte (pela ordem indicada):

- a diretoria onde se encontram as imagens
- número de *threads* a criar
- **-name** ou **-size** que indicam a ordem pela qual as imagens são organizadas antes de serem divididas pelas *threads*

como exemplificado de seguida.

```
./old-photo-parallel-B ./dir-1 4 -size  
./old-photo-parallel-B ./dir-2 8 -name  
./old-photo-parallel-B . 1 -name
```

A pasta onde se encontram as imagens pode ser uma relativa ao local onde o programa é executado (começando por **./**) ou absoluta se começar por **/**.

O número de *threads* deve ser um qualquer número inteiro positivo e indica quantas *threads* efetuarão o processamento das imagens. Por exemplo, se o utilizador indicar **1**


como o número de *threads* a criar, o programa utilizará apenas uma *thread* **para além** do *main()* para efetuar o processamento de todas as imagens.

O programa deverá processar todas as imagens cujo extensão é **.jpeg** que se encontrem na pasta indicada no primeiro argumento, ignorando todos os outros ficheiros.

Antes de criar a *threads*, o **main** deverá armazenar em memória o nome de todas as imagens **.jpeg** existente na pasta indicada. Os nomes dessas imagens deverão ser ordenados em memória por ordem alfabética (se o 3 argumento for **-name**) ou por tamanho (se o 3 argumento for **-size**). Só depois desta ordenação, se deverá proceder à divisão das imagens pelas *threads*.

Serão fornecidos conjuntos de imagens de modo que os alunos tenham dados variáveis. Os alunos podem naturalmente utilizar outras imagens durante o desenvolvimento e teste do projeto.

2.2 Impressão de estatísticas

Durante o processamento das imagens o utilizador poderá carregar na tecla **S/**  seguido de Enter/Return para saber quantas imagens já foram processadas e quantas faltam. Esta tecla de estatísticas também imprimirá o tempo médio de processamento das imagens já terminadas.

2.3 Resultados

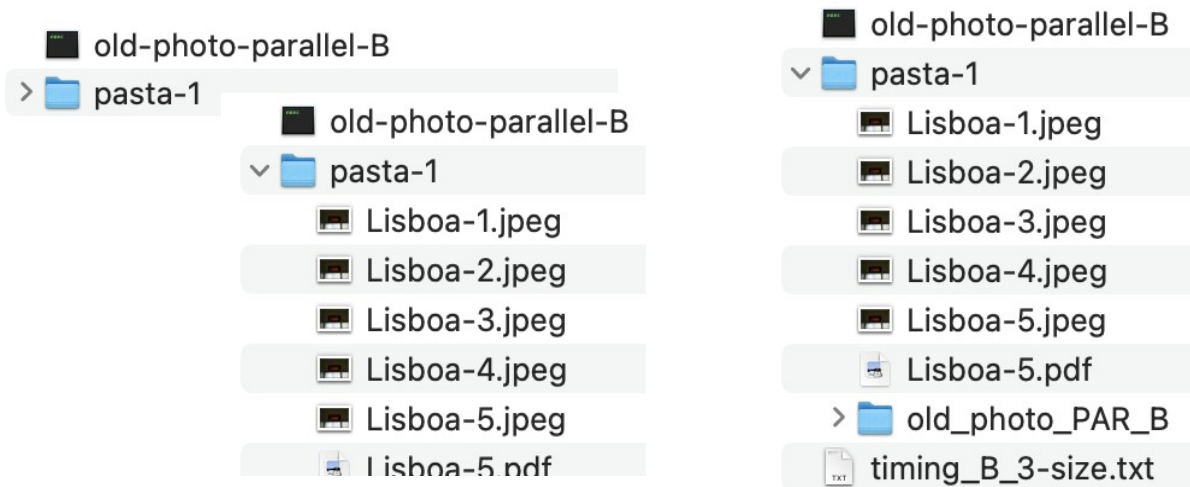
A execução da aplicação produz um conjunto de novas imagens, correspondentes à transformação de cada uma das imagens iniciais.

O nome das novas imagens será o mesmo da imagem original, mas colocadas numa diretoria específica e relativa à pasta indicada na linha de comandos. Essa diretoria dever-se-á chamar **old_photo_PAR_B** e deverá ser criada pela aplicação a

desenvolver na diretoria indicada na linha de comandos pelo utilizador onde se encontram as imagens originais.

As imagens seguintes exemplifica, a estrutura dos ficheiros e pastas antes (esquerda) e depois (direita) de executar a aplicação com os seguintes argumentos:

```
./old-photo-parallel-B ./pasta-1 3 -size
```



2.4 Interrupção de execução

Se a aplicação for interrompida a meio do processamento das imagens, apenas parte dos resultados terão sido produzidos e guardados no disco.

Se o utilizador voltar a executar a aplicação, não deverá ser necessário voltar a produzir os ficheiros resultado já existentes. A aplicação só deverá processar e gastar tempo na criação dos ficheiros em falta.

Para verificar se um ficheiro existe os alunos poderão usar as funções `access()` como no seguinte exemplo:

```
if( access( nome_fich, F_OK ) != -1){
    printf("%s encontrado\n", nome_fich);
}else{
    printf("%s nao encontrado\n", nome_fich);
}
```

3 Avaliação de desempenho.

O relatório a produzir pelos alunos no final do trimestre deverá conter os resultados da avaliação de desempenho (tempos de execução e *speedups*) do programa e de cada uma das *threads*. Para tal, será necessário instrumentar o código para recolher os tempos de processamento do programa e todas as *threads*.

Os alunos deverão incluir no código do **old-photo-parallel-B** as funções de leitura de tempos (como no laboratório 3) e em cada execução produzir um ficheiro com esses resultados:

- tempo total de execução
- tempo de execução de cada *thread*
- tempo de execução da parte não paralelizada

Este ficheiro dever-se-á chamar **timing_B_n-size.txt** ou **timing_B_n-name.txt** e deverá ser criado na pasta onde se encontram as imagens. O **n** deverá ser substituído pelo número de *threads* e **-size** ou **-name** dependerá do último argumento da linha de comandos.

4 Relatório

Com base nas duas aplicações desenvolvidas (**old-photo-parallel-A** e **old-photo-parallel-B**) os alunos deverão efetuar uma série de testes de desempenho (*speedups* atingidos) em dois computadores diferentes.

Com base nestes resultados os alunos deverão escrever um pequeno relatório que relate as descobertas e conclusões acerca das características dos computadores e efeitos das mesmas nos tempos de execução das diversas aplicações.

O modelo do relatório e *datasets* a usar serão fornecidos posteriormente.

5 Submissão do projeto

5.1 Prazo de submissão

O prazo para submissão da resolução da Parte A do projeto é dia **10 de Janeiro de 2025 às 19h00** no FENIX.

Antes da submissão, os alunos devem criar grupos de dois alunos e registá-los no FENIX.

5.2 Ficheiros a submeter

Os alunos deverão submeter um ficheiro **.zip** contendo:

- todo o código da aplicação **old-photo-parallel-B**,
- a `Makefile` para a compilação da aplicação **old-photo-parallel-B**,
- relatório,
- ficheiros de resultados usados no relatório.

Não incluir no .zip as imagens dos *datasets* utilizadas.

6 Avaliação do projeto

A nota para esta parte do projeto será dada tendo em consideração o seguinte:

- Funcionalidades implementadas
- Modo de gestão das *threads* e recursos
- Estrutura e organização do código
- Tratamento de erros
- Comentários
- Relatório