Programação Concorrente – 24/25
Relatório – Projeto

Grupo nº110 Rodrigo Pereira nº100080 João Lopes nº99973

1 Objetivos do trabalho

O trabalho proposto pretende a aprendizagem de como funcionam Threads, pipes e Mutexes, como os utilizar na linguagem de programação C e entender como a arquitetura do computador influencia a velocidade relacionada com o número de threads. Isto é obtido através de um programa que aplica um filtro em imagens, onde as imagens são distribuídas por diferentes threads.

2 Funcionalidades implementadas

Table 1: Funcionalidades implementadas

	Não implementada	Com falhas	Totalmente correta
Argumentos da linha de comando			x
Leitura dos nomes das imagens			x
Criação da lista			x
Ordenação da lista			x
Criação pastas resultados			x
old-photo-paralelo-A			
Criação de *threads*			x
Distribuição trabalho pelas *threads*			x
Verificação ficheiro existentes			x
Produção imagens transformadas			x
Gestão de memória		x	
Produção de ficheiro de estatísticas			x
old-photo-paralelo-B			
Criação de *threads*			x
Envio das imagens para as *threads*			x
Verificação ficheiro existentes			x
Produção imagens transformadas			x
Gestão de memória			x
Produção de ficheiro de estatísticas			x
Impressão das estatísticas no ecrã			x
Terminação das *threads*			x

2.1 Descrição das funcionalidades com falhas

Gestão de memoria, apesar de estar implementada para o normal funcionamento do codigo, a libertação de memoria dinâmica não se encontra completamente implementada para quando o codigo terminar abruptamente (erro de alocação de memoria, erro ao abrir ficheiro erro ao escrever ficheiro etc…). Ocorre também um problema na primeira entrega onde devido a um free(), o programa realiza segmentation fault em situações especificas. Isto no entanto foi alterado para ser possível obter resultados para o relatório.

3 Organização do código e estrutura de dados

Nesta secção os alunos deverão apresentar todas as novas funções e estruturas de dados especialmente implementadas para as várias paralelizações, indicando aquilo que é comum às 3 aplicações ou específico para cada uma.

Deverão ser apresentadas (protótipo e pequena descrição) as novas funções implementadas.

Os alunos também deverão apresentar as estruturas de dados usadas para:

- armazenar a informação das imagens
- armazenar informação de tempos
- distribuição dos dados pelas *threads* (**old-photo-paralelo-A**)
- envio de informação pelas/para as *threads* (**old-photo-paralelo-B** e **old-photo-pipeline**).

Qualquer alteração ao modo como as imagens são processadas (mudança do código fornecido) também deverá ser descrita aqui.

1. old-photo-parallel-A

- void process_photos - função que processa as fotos
- int old_photo_final, organiza as imagens por threads e cria os threads

2. old-photo-parallel-B

- void *process_photos(void * arg) - Thread que processa as fotos
- void *input_reader(void * arg) - Thread que processa o input do usuário
- Struct thread_arg_t - LinkedList onde guardamos a informação necessária para cada thread processar as imagens
- Struct filenames - LinkedList que guarda todas as imagens processadas e o tempo de processamento

4 Partição do trabalho (old-photo-paralelo-A)

4.1 Algoritmo de partição

```
extra_image=n_images%nthreads
min_image=n_images/nthreads
start_index=0
end_index=0
for x in range(0,nthreads)
    end_index=end_index+min_image
    if x>extra_image
        end_index++

    thread_image_start=start_index
    thread_image_end=end_index

    start_index=end_index
```

Table 2: Distribuição de 10 imagens por 3 threads

	Imagens a ser processadas por cada *thread*			
	1ª imagem da *thread*	2ª imagem da *thread*	3ª imagem da *thread*	4ª imagem da *thread*
*thread*_0	IST-1.jpg	IST-2.jpg	IST-3.jpg	IST-4.jpg
*thread*_1	IST-5.jpg	IST-6.jpg	IST-7.jpg	
*thread*_2	IST-8.jpg	IST-9.jpg	IST-10.jpg	

4.2 Envio de informação para as *threads*

Nesta secção os alunos deverão descrever que informação é enviada do *main()* para cada *thread*. Os alunos deverão descrever a informação explicitamente enviada através do último argumento do *pthread_create()* e que outra informação cada *thread* acede.

Os alunos deverão descrever as variáveis globais acedidas pelas *threads* e de que modo cada uma acede a esses dados.

Cada thread acede a imagem de textura, a pasta final

5 Envio de trabalho (old-photo-paralelo-B)(betta)

5.1 Descrição das *threads* e *pipes*

- Temos as threads colocadas pelo utilizador, desde que seja um número entre 1 e o máximo de imagens, um thread extra que ficará a ver o input
- Existem 2 pipes. Um deles comunica com a thread que processa as imagens enviando o nome do ficheiro. O outro pipe funcionará como um aviso para terminar a thread que fica atenta ao input do utilizador.
- Cada thread tem acesso a um node diferente de uma Lista com toda a informação necessária para processar as imagens. A thread que processa o input receberá a LinkedList que tem a informação dos ficheiros guardada.
- Existe também um mutex que impede as threads de processamento de imagem alterar os seus nodes de forma a que seja tudo acessado sem causar problemas concorrentemente.

5.2 Variáveis globais

Como variáveis globais, temos somente os dois pipes que utilizamos e um mutex.

5.3 Terminação das threads

Após toda a informação relacionada com o processamento de imagens for enviado, a thread main() dá join de forma que espere por todos os processos terminarem. Temos também uma outra thread aberta, essa thread irá estar à espera do input do utilizador. Colocamos um select que irá estar à atento ao pipe file descriptor e também ao STIND_FILENO, quando fecharmos o pipe o select será avisado terminando também essa thread.

6 Resultados

6.1 Descrição do ambiente de execução

Nesta secção os alunos deverão descrever os dois computadores onde foram executados os testes:

➔ Computador 1

- Asus Tuff
- 16 cores
  - Cores reais ou virtuais
  - Cores de desempenho ou eficiência
- 3.2 GHz
- Windows Subsystem for Linux(WSL)

➔ Computador 2

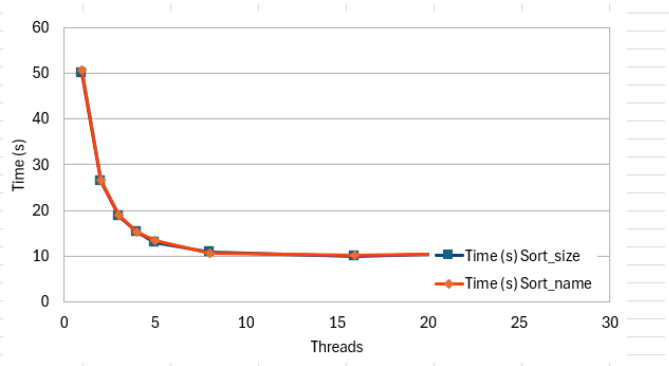
- OMEN Laptop 15

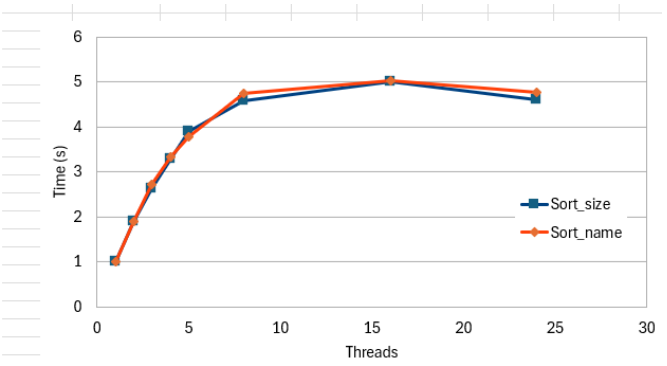
- AMD Ryzen 7 4800H 16 CPUs
- 2.9 GHz
- Windows Subsystem for Linux

6.2 Resultados old-photo-paralelo-A

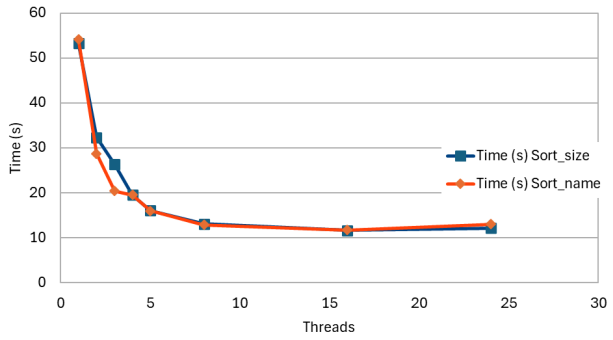
6.2.1 Dataset A

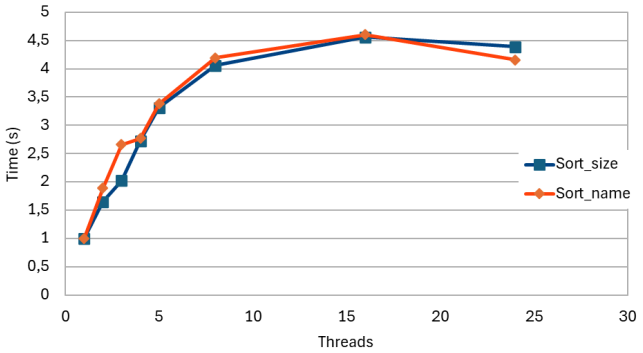
6.2.1.1 Computador 1





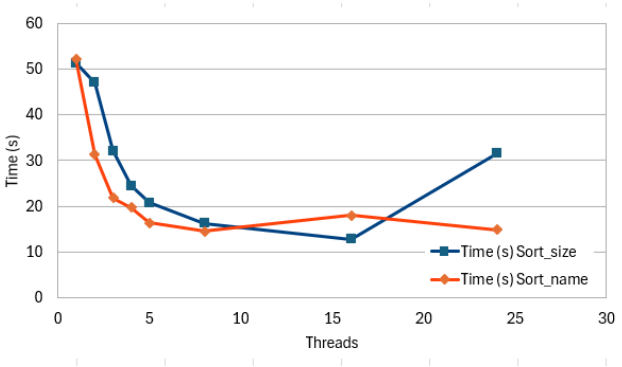
6.2.1.2 Computador 2

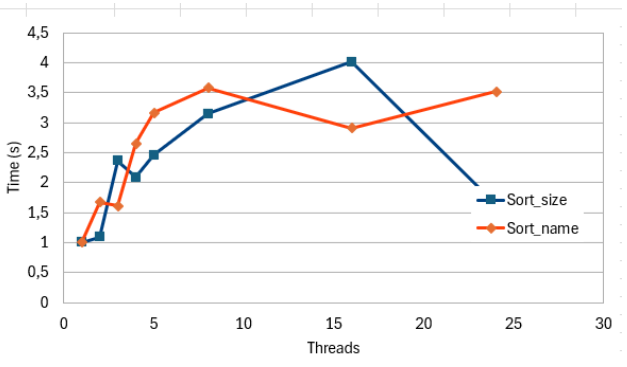




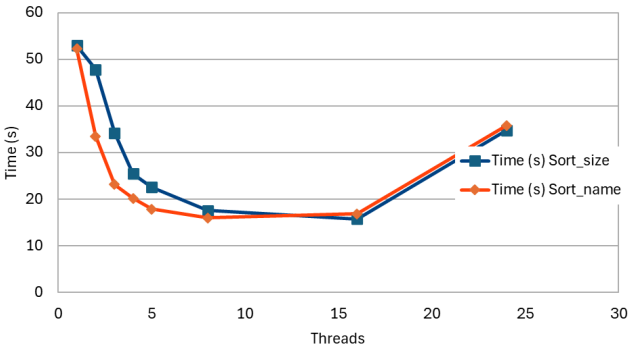
6.2.2 Dataset B

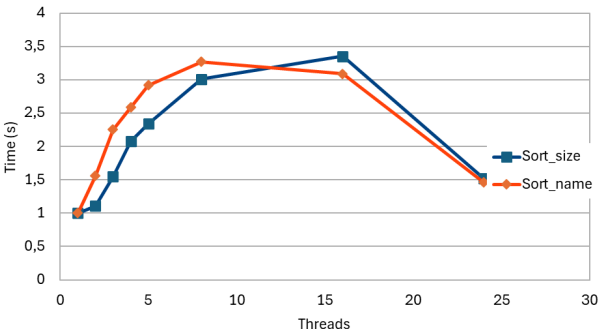
6.2.2.1 Computador 1





6.2.2.2 *Computador 2*

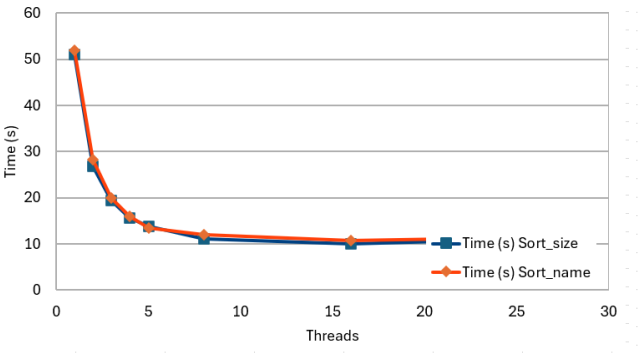


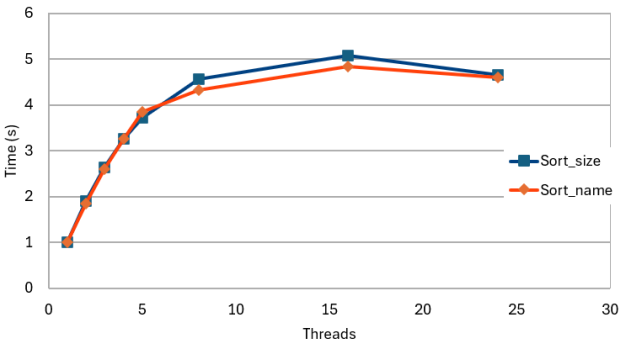


6.3 Resultados old-photo-paralelo-B

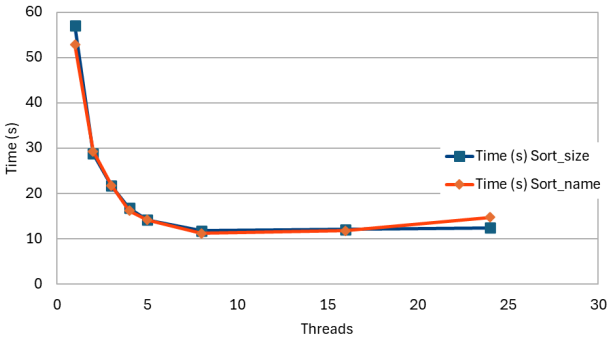
6.3.1 Dataset A

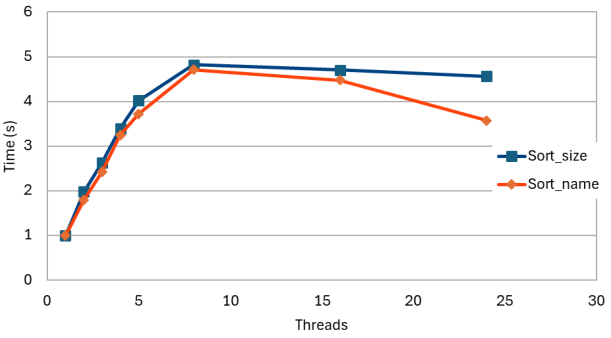
6.3.1.1 *Computador 1*





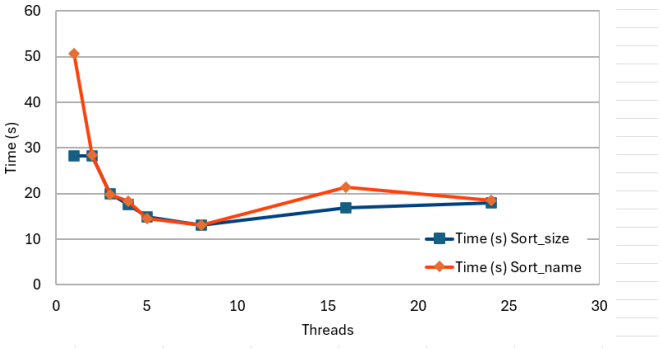
6.3.1.2 *Computador 2*

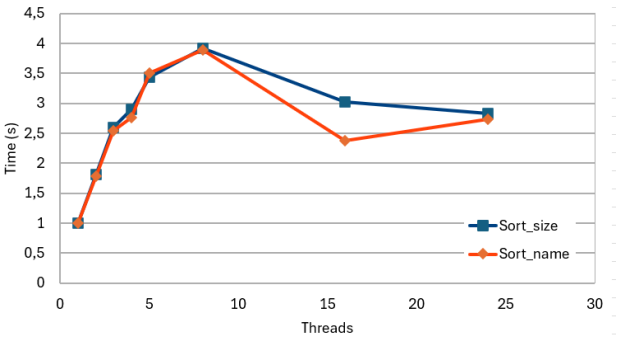




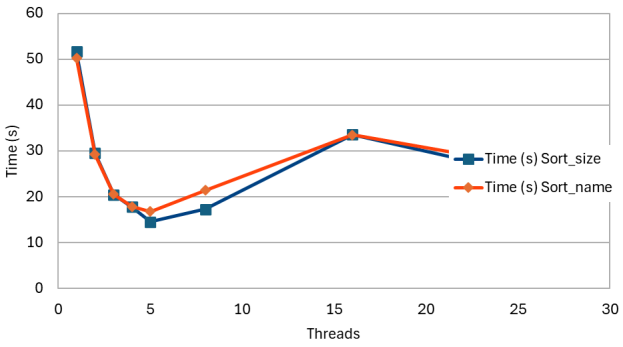
6.3.2 Dataset B

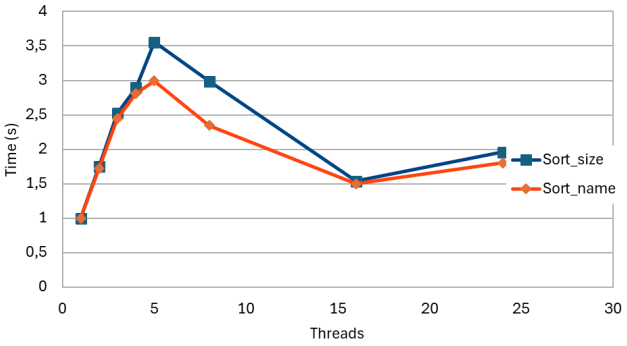
6.3.2.1 Computador 1





6.3.2.2 Computador 2





7 Discussão

7.1 Efeito do número de Cores

O aumento inicial de performance é significativo quando o número de threads aumenta até o número de cores físicos disponíveis, Após isso, o uso de threads adicionais traz poucos ganhos ou piora mesmo o resultado, apos ultrapassar o numero de cpus fisicos ele começa a criar cores virtuais.

Podemos ver isto no grafico 1 e no grafico 2 onde o tempo estabiliza e os speedups diminui

7.2 Efeito da ordem das imagens

Dependendo do algoritmo de distribuição de imagens os threads, quando damos sort por sized os threads iniciais ficam com as imagens mais leves e os threads finais ficam com as imagens mais pesadas, assim não fica bem distribuída o tempo de processamento.

7.3 Old-photo-paralelo-A vs old-photo-paralelo-B

Os resultados de old-photo-paralelo-B como podemos ver são batante parecidos aos de old photo paralelo A, pois o limite físico de threads continua a ser o maior bloqueador de performance, mas podemos ver que apesar disto eles conseguem atingir speedups ligeiramente maiores. isto deve se ao melhor algoritmo de distribuição das imagens e a rapidez de organização do código (agora gerido pelos pipes).

8 Conclusão

O projeto foi desenvolvido com sucesso e o principal objetivo foi alcançado.

Durante o desenvolvimento e a utilização das threads entendemos como estas estão dependentes dos cpus físicos e como as terminar de forma correcta.

As principais dificuldades foram mesmo no desenvolvimento e fechar da thread que acede à memória partilhada e olha para o teclado pois aqui encontramos algumas dificuldades iniciais tanto a fechá-la tanto para o correto funcionamento da mesma.

9 Anexo

Informações de Sistema

Hora/data atual:	10 de janeiro de 2025, 00:01:57
Nome do Computador:	LAPTOP-OPRS4DFS
Sistema Operativo:	Windows 10 Home 64 bits (10.0, Compilação 19045)
Idioma:	português (Definição regional: português)
Fabricante do Sistema:	ASUSTeK COMPUTER INC.
Modelo do Sistema:	ASUS TUF Gaming A15 FA506QM
BIOS:	FA506QM.314
Processador:	AMD Ryzen 7 5800H with Radeon Graphics (16 CPUs), ~3.2GHz
Memória:	16384MB RAM
Ficheiro de paginação:	23334MB utilizados, 9521MB disponíveis
Versão do DirectX:	DirectX 12

Informações de Sistema

Hora/data atual:	10 de janeiro de 2025, 14:47:19
Nome do Computador:	ERREENE
Sistema Operativo:	Windows 11 Home 64 bits (10.0, Compilação 22631)
Idioma:	português (Definição regional: português)
Fabricante do Sistema:	HP
Modelo do Sistema:	OMEN Laptop 15-en0xxx
BIOS:	F.07
Processador:	AMD Ryzen 7 4800H with Radeon Graphics (16 CPUs), ~2.9GHz
Memória:	16384MB RAM
Ficheiro de paginação:	20896MB utilizados, 8660MB disponíveis
Versão do DirectX:	DirectX 12