



UNICAMP UNIVERSIDADE ESTADUAL DE CAMPINAS

---

# UNIVERSIDADE ESTADUAL DE CAMPINAS

**Faculdade de Tecnologia - FT**

SISTEMAS OPERACIONAIS

PROF. ANDRÉ LEON SAMPAIO GRADVOHL

PROJETO FINAL SISTEMAS OPERACIONAIS

Projeto\_1

---

Grupo Inventado

Nome:

RA:

Nikolas D'Andrea Freire

222970

Victor Akio Uchiyama

244974

Wallace Benedito da Silva

245752



---

Limeira - Novembro de 2019

## Sumário

<b>1. Repositório GIT3.....</b>	<b>3</b>
1.1 Link GitHub.....	3
1.2 Link Vídeo.....	3
<b>2. O problema.....</b>	<b>4</b>
2.1 Descrição da solução do problema.....	5
2.2 Algoritmo em alto nível.....	6
2.3 Instruções de compilação.....	7
<b>3. Gráficos de tempo de execução.....</b>	<b>8</b>
<b>4. Conclusão.....</b>	<b>9</b>

## 1. REPOSITÓRIO GIT

Segue o link para o repositório GitHub com toda a documentação do projeto, assim como o código fonte documentado e o link para acesso ao vídeo de execução do programa.

### 1.1 Link GitHub

<https://github.com/akiouchiyama/grupoinventadoSO>

### 1.2 Link vídeo

<https://www.youtube.com/watch?v=AJ1fzDdR2YM&feature=youtu.be>

## 2. O PROBLEMA

Foi proposto para a nossa equipe, elaborar um programa que utilize múltiplas threads para ler valores inteiros de N arquivos de entrada, armazenando esses valores em ordem crescente em um único arquivo de saída.

A utilização de múltiplas threads teve como objetivo a quantificação de desempenho operacionais da execução desse sistema, utilizando a unidade de tempo (segundos e milissegundos) para cada combinação de threads. A partir desse contexto foi possível gerar um relatório de desempenho e de análise operacional.

## 2.1 Descrição da Solução do Problema

O projeto criado pela equipe *Inventado* conseguiu introduzir a função de múltiplas threads (2, 4, 8, 16) para ordenar uma sequência de números inteiros em múltiplos arquivos e posteriormente ordená-los num único arquivo.

Nossa lógica para a resolução do problema foi a seguinte: separar etapas do projeto em funções diferentes como Leitura dos diferentes arquivos realizado sem o uso de threads, ordenação multithread e escrita do arquivo final sem o uso de threads, onde a main realiza a operação de apenas receber os argumentos da linha de comando, criar as threads chamar a função leitura que lê todos os arquivos e salva em um único vetor e a contagem do tempo com a função clock(), e enviar cada thread para a função ordena com um laço o que cada thread fará é ordenar o vetor como passamos o endereço do vetor ela simultaneamente ordenara este, por fim recebemos o tempo final que levamos para realizar esta operação e enviamos o vetor ordenado para a função escreve que imprime o vetor no arquivo de saída.

## 2.2 Instruções de compilação

Para a compilação é necessário acessar o link do repositório no GitHub, realizar o download em <https://github.com/akiouchiyama/grupoinventadoSO>, em seguida, acessar via terminal do sistema operacional Linux a pasta no qual estão os arquivos baixados e inserir os seguintes comandos:

```
gcc -pthread thread.c -o thread
```

Em seguida digite:

```
./thread numero_de_threads arqN.txt saida.txt
```



Segue abaixo a ilustração da compilação do programa:

<code>./antigo 2 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt</code>	<b>//PARA 2 THREADS</b>
<code>./antigo 4 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt</code>	<b>//PARA 4 THREADS</b>
<code>./antigo 8 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt</code>	<b>//PARA 8 THREADS</b>
<code>./antigo 16 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt</code>	<b>//PARA 16 THREADS</b>

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
victor@victor-Lenovo-ideapad-330-15IKB:~/Documentos/sistemas_operacionais/antigo$ ./antigo 2 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt
Tempo de processamento usando 2 threads: 0.329936 segundos
victor@victor-Lenovo-ideapad-330-15IKB:~/Documentos/sistemas_operacionais/antigo$ ./antigo 4 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt
Tempo de processamento usando 4 threads: 0.600704 segundos
victor@victor-Lenovo-ideapad-330-15IKB:~/Documentos/sistemas_operacionais/antigo$ ./antigo 8 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt
^[[ATempo de processamento usando 8 threads: 1.540014 segundos
victor@victor-Lenovo-ideapad-330-15IKB:~/Documentos/sistemas_operacionais/antigo$ ./antigo 16 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt
Tempo de processamento usando 16 threads: 3.110608 segundos
victor@victor-Lenovo-ideapad-330-15IKB:~/Documentos/sistemas_operacionais/antigo$
```

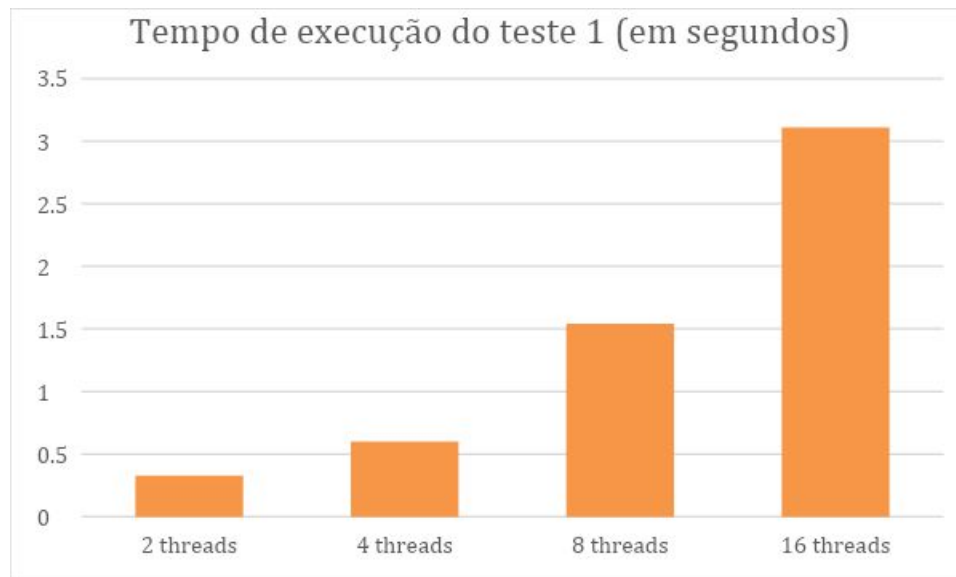
**Obs:** Como o programa acima foi executado em um dos computadores dos membros da equipe, o nome do código fonte está diferente de como está no GitHub. Portanto, no momento da compilação, ao invés de digitar “./antigo”, digite “./thread”.



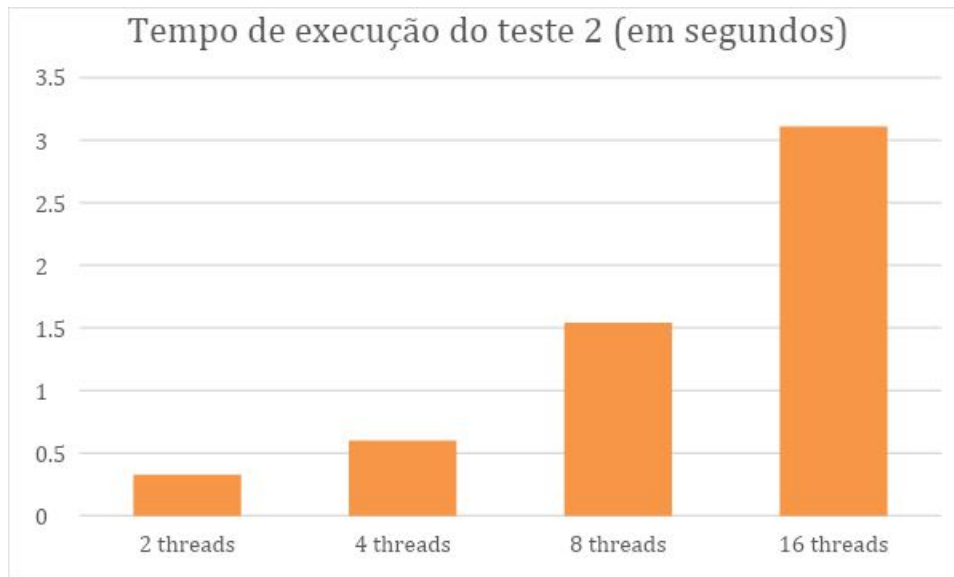
### 3. GRÁFICOS DE EXECUÇÃO

Para este teste foi utilizado uma máquina com processador Intel Core i7-8550U de 8ª Geração (1.8GHz até 4.0GHz) 8Mb Cache, Memória RAM 8GB DDR4.

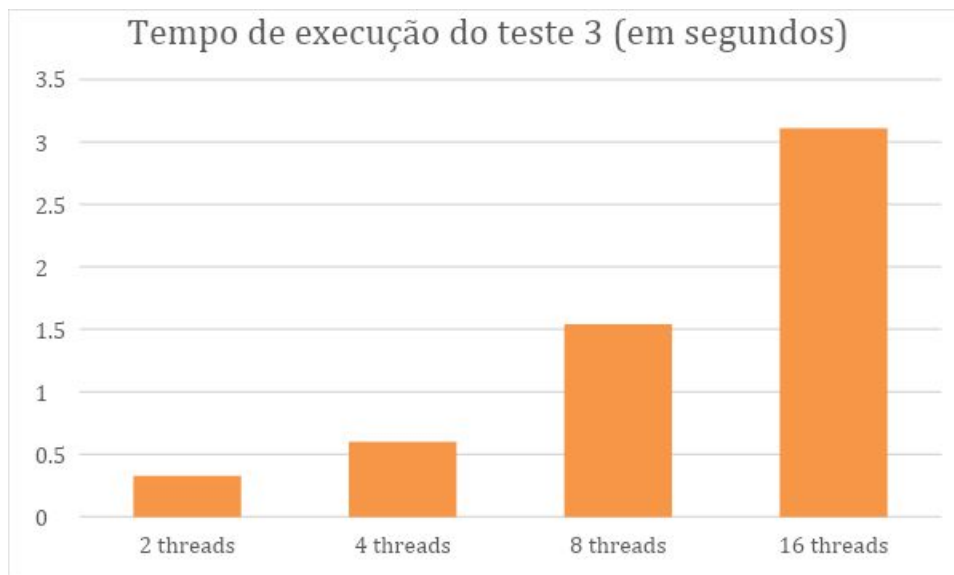
Teste 1:



Teste 2:

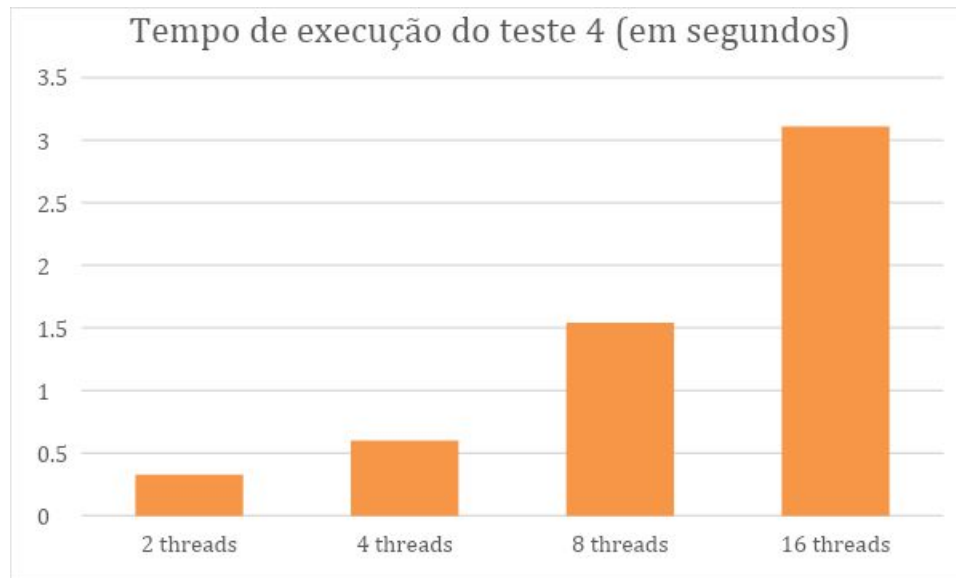


Teste 3:





Teste 4:





#### 4. CONCLUSÃO

Durante o processamento de um programa, ele é sub-dividido em várias linhas de execução, possuindo uma ordem de atendimento. Essas linhas têm o processamento dividido pelo sistema operacional e repassa ao processador. É basicamente assim que funciona o conceito de multi – thread (**podemos achar na net, alguma definição melhor**).

Com base nos testes feitos pelo grupo, ao se executar um programa com uma grande variedade de dados é benéfico utilizar múltiplas threads, e até possibilita a melhoria do desempenho conforme o número de threads vai aumentando. No entanto, ao se executar pequenas quantias de dados utilizando conceitos de múltiplas threads poderá perder a eficiência já que o essa sub-divisão das tarefas demanda de processamento. (**dar uma melhorada nisso e modificar pra não ficar muito igual ao do colega**) .