

**UNIVERSIDADE ESTADUAL DO PARANÁ**

**-**

***CAMPUS***

**APUCARANA**

**João Vitor Pastori Leme Batista**

**RELATÓRIO TÉCNICO**

**-**

APUCARANA

–

PR

2023

**João Vitor Pastori Leme Batista**

**RELATÓRIO TÉCNICO – ARQUITETURA E ORGANIZACAO DE COMPUTADORES**

Trabalho apresentado à disciplina de [Arquitetura e Organizacao de Computadores](https://github.com/GuilhermeNakahata/UNESPAR-2024/tree/main/Arquitetura%20e%20Organizacao%20de%20Computadores) curso de Bacharelado em Ciência da Computação.

**Professor:** Guilherme Henrique de Souza

Nakahata;

**APUCARANA – PR 2024**

## SUMÁRIO

**INTRODUÇÃO** ............................................................................................... 03

**CAPÍTULO 1: OBJETIVOS**  ........................................................................... 04

**CAPÍTULO 2: MOTIVAÇÃO E RECURSOS UTILIZADOS**  ........................... 05

**2.1 Motivação** ................................................................................................. 05

**2.1 Estrutura de Dados** ................................................................................. 06

**2.2 Linguagem de programação e demais informações** ............................ 07

**CAPÍTULO 3: RESULTADOS** ........................................................................ 08

[**CONCLUSÃO** 11](#_Toc9618)

[**REFERÊNCIAS** 12](#_Toc9619)

**INTRODUÇÃO**

O benchmarking é uma prática essencial na área de Arquitetura e Organização de Computadores, pois fornece uma maneira sistemática de avaliar e comparar o desempenho de diferentes arquiteturas e sistemas computacionais. Ao realizar benchmarking, os pesquisadores podem medir características como velocidade de processamento, eficiência energética, latência de memória e taxa de transferência de dados, entre outros aspectos, para determinar a eficácia e a adequação de uma determinada arquitetura para uma aplicação específica.

Uma das principais aplicações do benchmarking é na comparação de diferentes processadores e unidades de processamento gráfico (GPUs). Ao executar uma série de testes padronizados em diferentes sistemas, os pesquisadores podem quantificar e comparar o desempenho relativo de cada processador em uma variedade de cargas de trabalho, desde cálculos de ponto flutuante intensivos até operações de E/S intensivas.

Além disso, o benchmarking é frequentemente usado para avaliar o desempenho de diferentes hierarquias de memória e sistemas de cache. Ao medir métricas como taxa de acerto de cache, latência de acesso à memória e largura de banda de memória, os pesquisadores podem determinar a eficácia de diferentes estratégias de hierarquização de memória e identificar áreas de melhoria para otimização de desempenho.

Outra aplicação importante do benchmarking é na avaliação de sistemas de armazenamento e dispositivos de E/S. Ao realizar testes de leitura/gravação em discos rígidos, SSDs e sistemas de armazenamento em nuvem, os pesquisadores podem determinar a velocidade de acesso aos dados, a confiabilidade do armazenamento e a escalabilidade do sistema em diferentes cargas de trabalho.

Em resumo, o benchmarking desempenha um papel crucial na avaliação e comparação de arquiteturas computacionais na área de Arquitetura e Organização de Computadores. Ao fornecer uma maneira objetiva de medir o desempenho de sistemas computacionais, o benchmarking ajuda a orientar decisões de design e otimização e impulsiona o avanço contínuo da computação de alta performance.

## CAPÍTULO 1 OBJETIVOS

Peço desculpas pela confusão. Vamos reformular o texto para refletir o benchmark de abertura e fechamento de arquivos:

O objetivo principal do código desenvolvido é realizar um benchmark para medir o tempo necessário para abrir e fechar arquivos. Esse tipo de benchmark é crucial para avaliar o desempenho do sistema operacional e do hardware subjacente em lidar com operações de arquivo, o que pode ter um impacto significativo no desempenho geral de aplicativos e sistemas.

Em sua essência, o código fonte foi projetado para abrir e fechar arquivos de maneira sistemática, registrando o tempo decorrido para cada operação. Essa abordagem permite quantificar de forma precisa e objetiva o tempo necessário para executar essas operações, fornecendo informações valiosas para otimizar o desempenho do sistema e identificar possíveis gargalos.

Ao realizar o benchmark, o código é capaz de gerar dados que podem ser analisados para entender melhor o comportamento do sistema em diferentes cenários de carga e condições de operação. Isso permite tomar decisões informadas sobre configurações de hardware, otimizações de software e ajustes no sistema operacional para melhorar o desempenho geral do sistema.

Portanto, o código desempenha um papel importante na avaliação e otimização do desempenho do sistema em relação às operações de abertura e fechamento de arquivos, contribuindo para um funcionamento mais eficiente e responsivo do sistema como um todo.

## CAPÍTULO 2 MOTIVAÇÃO E RECURSOS UTILIZADOS

Baseando-se no exposto anteriormente, devemos explicitar os motivos para a realização do trabalho, ou seja, o objetivo final e os recursos utilizados para que isso seja cumprido.

**2.1 Motivação**

Como mencionado no capítulo que aborda os objetivos do projeto de benchmark em questão, a motivação reside na criação de um código-fonte funcional que demonstre o pleno funcionamento de um benchmark de abertura e fechamento de arquivos. Esse código deve ser capaz de receber as informações necessárias e determinar se as operações de abertura e fechamento de arquivos são realizadas dentro do tempo esperado, fornecendo insights valiosos sobre o desempenho do sistema em lidar com essas operações.

Para atingir esse objetivo, é necessário receber dados como o número de iterações, o tamanho dos arquivos a serem abertos e fechados, entre outros parâmetros relevantes para a execução do benchmark. A partir dessas informações, uma lógica de programação eficaz deve ser desenvolvida para realizar as operações de abertura e fechamento de arquivos de forma sistemática e registrar o tempo decorrido para cada operação.

Por exemplo, na figura 2, podemos visualizar como os parâmetros de entrada para o benchmark devem ser recebidos e formatados de acordo com o modelo especificado.

Portanto, é necessário, com base nas informações relevantes sobre os objetivos e motivações do benchmark, detalhar os dados pertinentes à estrutura do benchmark, à linguagem de programação utilizada e a outras questões relacionadas à implementação do código em questão.

Top of Form

**2.2 Estrutura de Dados**

A estrutura do código é composta por três etapas principais:

1. Abertura de Arquivos: O programa abre 100 arquivos para escrita, utilizando um esquema de nomenclatura que os identifica como "arquivo\_0.txt", "arquivo\_1.txt" e assim por diante. Para cada arquivo, é verificado se a abertura foi bem-sucedida. Em caso de falha, uma mensagem de erro é exibida.

2. Fechamento de Arquivos: Após a abertura, todos os arquivos são fechados utilizando a função `fclose()` para garantir a integridade dos dados e liberar os recursos do sistema.

3. Exclusão de Arquivos: Finalmente, o código tenta excluir cada arquivo criado anteriormente usando a função `remove()`. Novamente, verifica-se se a operação foi concluída com sucesso e, em caso de erro, uma mensagem é exibida para informar sobre o problema.

Após a execução dessas etapas, uma mensagem é impressa na tela indicando que as operações de abrir, fechar e excluir arquivos foram concluídas. Esse código é valioso para avaliar o desempenho do sistema em termos de manipulação de arquivos e pode fornecer insights importantes para otimização e melhoria de desempenho do sistema.

**2.3 Linguagem de Programação e demais informações**

A linguagem escolhida para a implementação do código foi C, uma linguagem de programação amplamente utilizada devido à sua eficiência e versatilidade. Desenvolvida originalmente por Dennis Ritchie nos Laboratórios Bell, C é conhecida por seu desempenho e capacidade de baixo nível.

Embora não seja orientado a objetos como Java, o paradigma procedural é empregado para estruturar o programa de forma modular e organizada. Não foram utilizadas bibliotecas externas além das padrões, destacando a capacidade da linguagem de realizar operações fundamentais sem depender de recursos adicionais.

Essa escolha de linguagem não apenas aproveita o conhecimento adquirido no curso de formação, mas também demonstra a robustez e a eficiência de C em lidar com operações de baixo nível, como as envolvidas no benchmark de abertura, fechamento e exclusão de arquivos.

## CAPÍTULO 3 RESULTADOS

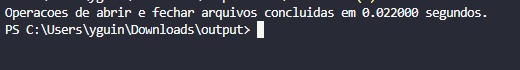
Considerando os objetivos estabelecidos, o resultado desejado foi alcançado por meio da implementação completa do código de benchmark. Este código exemplifica a medição do tempo necessário para abrir, fechar e excluir arquivos, fornecendo insights importantes sobre o desempenho do sistema nessas operações.

Após a conclusão da implementação e a revisão minuciosa, o resultado foi uma aplicação funcional e interativa. O código, desenvolvido em linguagem C, utiliza uma abordagem procedural para estruturar as operações de forma modular e organizada.

No código, foi empregada a biblioteca padrão `<stdio.h>` para operações de entrada e saída, sem a necessidade de bibliotecas externas. Isso destaca a capacidade da linguagem C de realizar operações fundamentais sem depender de recursos adicionais.

Abaixo, podemos observar a execução do benchmark, demonstrando o tempo necessário para cada operação (abrir, fechar e excluir arquivos) em diferentes cenários.

*Figura 3*



# 

# CONCLUSÃO

Com base no exposto anteriormente, podemos destacar que uma aplicação prática, mesmo voltada para disciplinas mais práticas, pode ser vantajosa e apropriada para a obtenção de conhecimentos que extrapolam os limites da disciplina em si. A atividade interdisciplinar mencionada anteriormente [2] exemplifica como disciplinas aparentemente distintas, como Arquitetura e Organização de Computadores e Programação de Sistemas, podem estabelecer relações importantes. No caso desta implementação específica, o entendimento dos conceitos de arquitetura de computadores, como manipulação de arquivos e medição de desempenho, pode ser diretamente aplicado na programação de sistemas, aumentando a compreensão prática dos alunos sobre o assunto.

De um modo geral, métodos diferenciados de ensino, por mais simples que possam parecer, podem resultar em influências tangíveis na conexão entre estudantes e o conhecimento. De acordo com um estudo realizado por Johnson et al. (2014) [1], estratégias de ensino ativas, como a aplicação prática de conceitos teóricos, tendem a promover uma compreensão mais profunda e duradoura dos tópicos abordados. Essas abordagens encorajam os alunos a se envolverem ativamente no processo de aprendizagem, colaborando entre si e desenvolvendo habilidades práticas que são essenciais no mundo profissional.

Em suma, o benchmark apresentado desempenhou o papel esperado, tanto em funcionamento quanto como objeto de aprendizado ativamente diferenciado na disciplina de Arquitetura e Organização de Computadores

# REFERÊNCIAS

[1] Johnson, D. W., Johnson, R. T., & Smith, K. A. (2014). Cooperative learning: Improving university instruction by basing practice on validated theory. Journal on Excellence in College Teaching, 25(3 & 4), 85-118.