



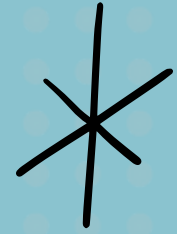
EDITOR DE TEXTO

Utilizando
5 threads



Apresentado por Andersson Silva e Andreza Gonçalves





REQUISITOS



Em editores de texto.



REQUISITO 1

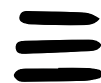
Ler um arquivo;

REQUISITO 2

Contar letras, números, caracteres especiais e quantidade de linhas;

REQUISITO 3

Criar arquivos para guardar os backups do requisito anterior.



SOLUÇÃO 1

Uma thread vai ler o arquivo e contar os caracteres;



SOLUÇÃO 2

4 threads para receber os dados das thread anteriores para escrever em arquivos;



SOLUÇÃO 3

Uso da linguagem Rust para melhor eficiência.



PROPOSTA

Propostas para melhorias ideais.



O PROGRAMA

Melhor editor de texto de Roraima.



PROGRAMA

```
função main() {  
  
};
```

▶ Run | ⌘ Debug

```
fn main() {  
    let nome_arquivo: &'static str = "texto.txt";  
    println!("nome do arquivo: {}", nome_arquivo);  
  
    let handle: JoinHandle<()> = thread::spawn(move || {  
        ler_arquivo(nome_arquivo);  
    });  
  
    // Aguarda a thread terminar  
    handle.join().expect(msg: "Thread principal falhou");  
}
```



PROGRAMA

função ler_arquivo() {

};



PROGRAMA

função ler_arquivo() {

};

```
for linha: Result<String, Error> in reader.lines() {  
  match linha {  
    Ok(l: String) => {  
      numero_linhas += 1;  
      for caractere: char in l.chars() {  
        if caractere.is_ascii_digit() {  
          numeros.push(caractere);  
        } else if caractere.is_alphabetic() {  
          letras.push(caractere);  
        } else {  
          caracteres_especiais.push(caractere);  
        }  
      }  
    }  
    Err(e: Error) => println!("deu erro ai: {}", e),  
  }  
}
```

loop para ler todos as linhas e
caracteres



PROGRAMA

função ler_arquivo() {

};

```
let retorno_scope: i32 = thread::scope(|scope: &Scope<'_, '_>| {  
    let mut handles: Vec<ScopedJoinHandle<'_, ...>> = Vec::new();  
  
    handles.push(scope.spawn(|| {  
        println!("thread 01 - numeros");  
        backup_numeros(&numeros);  
    }));  
  
    handles.push(scope.spawn(|| {  
        println!("thread 02 - letras");  
        backup_letras(vetor_letra: &letras);  
    }));  
  
    handles.push(scope.spawn(|| {  
        println!("thread 03 - caracter especial");  
        backup_caracter_especial(vetor_caracter: &caracteres_especiais);  
    }));  
  
    handles.push(scope.spawn(|| {  
        println!("thread 04 - numeros de linhas");  
        backup_numero_linha(numero_linhas);  
    }));  
});
```

criação das threads para escrita



PROGRAMA

função ler_arquivo() {

};

```
let mut total_thread_finalizada: i32 = 0;  
  
for thread_num: ScopedJoinHandle<'_, ()> in handles.into_iter() {  
    if let Ok(_thread_finalizada: ()) = thread_num.join() {  
        total_thread_finalizada += 1;  
    }  
}  
  
total_thread_finalizada  
});  
  
println!("total de theads: {}", retorno_scope);  
} fn ler_arquivo
```

loop de espera para finalização
das threads e encerramento do
programa



PROGRAMA

função backup_letras() {

};

```
fn backup_letras(vetor_letra: &Vec<char>) {  
    let nome_arquivo_backup_letras: &'static str = "backup_letras.txt";  
  
    let mut arquivo_backup_letras: File = match File::create(path: nome_arquivo_backup_letras) {  
        Ok(f: File) => {  
            println!("Arquivo criado com sucesso");  
            f  
        }  
  
        Err(e: Error) => {  
            println!("Deu ruim na hora de criar o arquivo: {}", e);  
            process::exit(code: 1)  
        }  
    };  
  
    for letra: &char in vetor_letra {  
        if let Err(e: Error) = write!(arquivo_backup_letras, "{}", letra) {  
            println!("deu BO mano: {}", e);  
        }  
    }  
  
    println!("Fim da thread 02");  
}
```

função para escrever as letras em
um arquivo de backup



PROGRAMA

função backup_numeros() {

};

```
fn backup_numeros(numeros: &Vec<char>) {  
    let nome_arquivo_backup_numeros: &'static str = "backup_numeros.txt";  
  
    let mut arquivo_backup_numeros: File = match File::create(path: nome_arquivo_backup_numeros) {  
        Ok(f: File) => {  
            println!("Arquivo criado com sucesso");  
            f  
        }  
  
        Err(e: Error) => {  
            println!("Deu ruim na hora de criar o arquivo: {}", e);  
            process::exit(code: 1)  
        }  
    };  
  
    for numero: &char in numeros {  
        if let Err(e: Error) = write!(arquivo_backup_numeros, "{}", numero) {  
            println!("deu BO mano: {}", e);  
        }  
    }  
  
    println!("Fim da thread 01");  
}
```

função para escrever os números
em um arquivo de backup



PROGRAMA

função
backup_numero_linha() {

};

```
fn backup_numero_linha(numero_linhas: usize) {  
    let nome_arquivo_backup_numeros_linhas: &'static str = "backup_numeros_linhas.txt";  
  
    let mut arquivo_backup_numeros_linhas: File = match File::create  
    (path: nome_arquivo_backup_numeros_linhas) {  
        Ok(f: File) => {  
            println!("Arquivo criado com sucesso");  
            f  
        }  
  
        Err(_e: Error) => {  
            println!("Deu ruim na hora de criar o arquivo");  
            process::exit(code: 1)  
        }  
    };  
  
    if let Err(e: Error) = write!(arquivo_backup_numeros_linhas, "{}", numero_linhas) {  
        println!("deu ruim: {}", e);  
    }  
  
    println!("Fim da thread 04");  
}
```

função para escrever o número de
linhas em um arquivo de backup



PROGRAMA

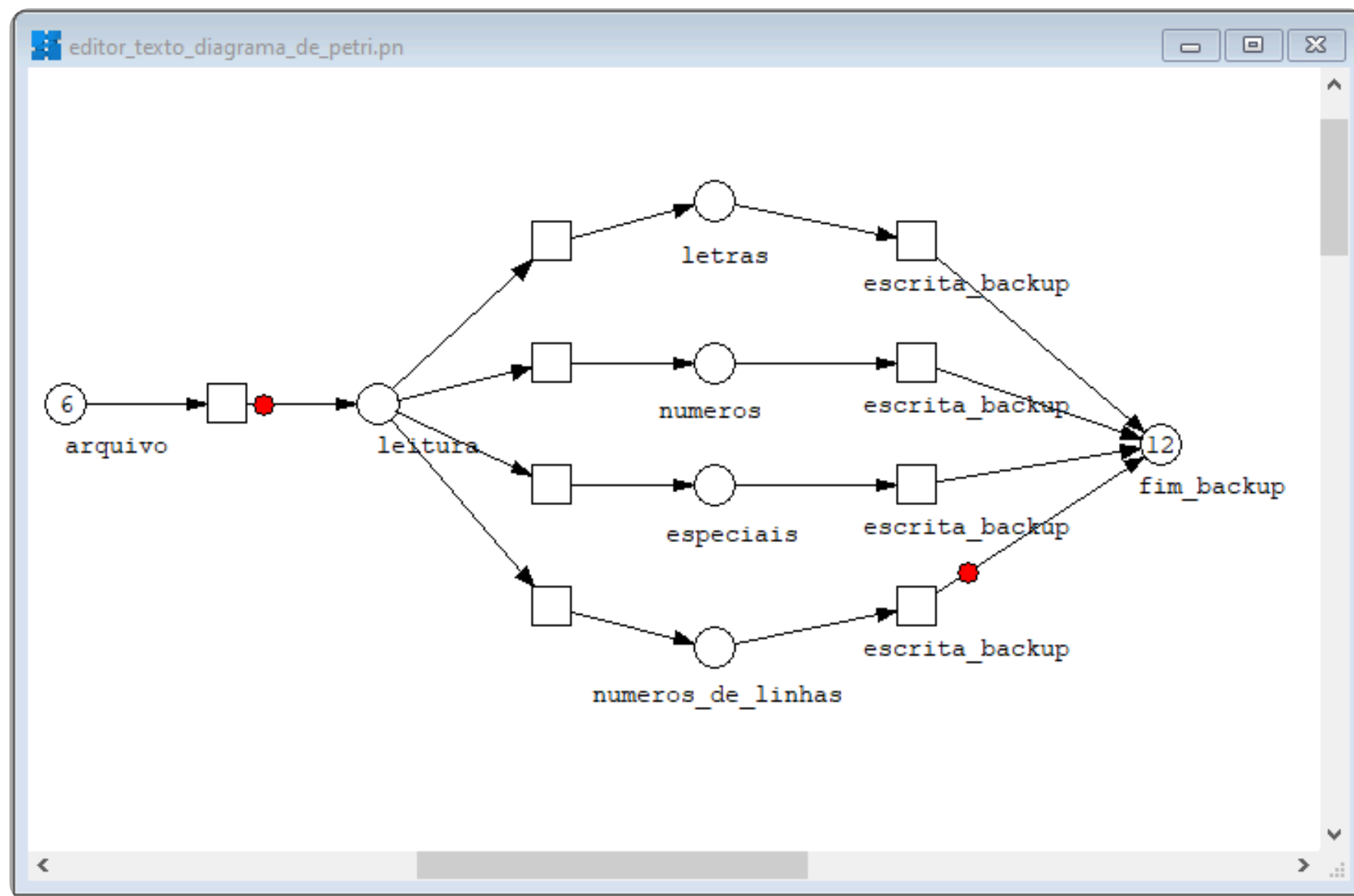
função

```
backup_caracter_especial() {  
  
};
```

```
use std::fs::File;  
use std::io::{BufRead, BufReader, Write};  
use std::{process, vec};  
use std::thread;  
  
fn backup_caracter_especial(vetor_caracter: &Vec<char>){  
    let nome_arquivo_backup_caracter_especial: &'static str = "backup_caracter_especial.txt";  
  
    let mut arquivo_backup_caracter_especial: File = match File::create  
    (path: nome_arquivo_backup_caracter_especial) {  
        Ok(f: File) => {  
            println!("Arquivo criado com sucesso");  
            f  
        }  
  
        Err(e: Error) => {  
            println!("Deu ruim na hora de criar o arquivo: {}", e);  
            process::exit(code: 1)  
        }  
    };  
  
    for caractere: &char in vetor_caracter {  
        if let Err(e: Error) = write!(arquivo_backup_caracter_especial, "{}", caractere) {  
            println!("deu BO mano: {}", e);  
        }  
    }  
  
    println!("\nFim da thread 03")  
}
```

importação das bibliotecas e e função de criação
de um arquivo backup para caracteres especiais

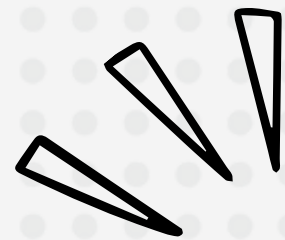
DIAGRAMA DE PETRI



Modelado no Snoopy.

- Lê um arquivo de texto;
- Para cada unidade lida, executa classificações simultâneas;
- O paralelismo é modelado com todas as threads iniciando a partir de uma leitura central;

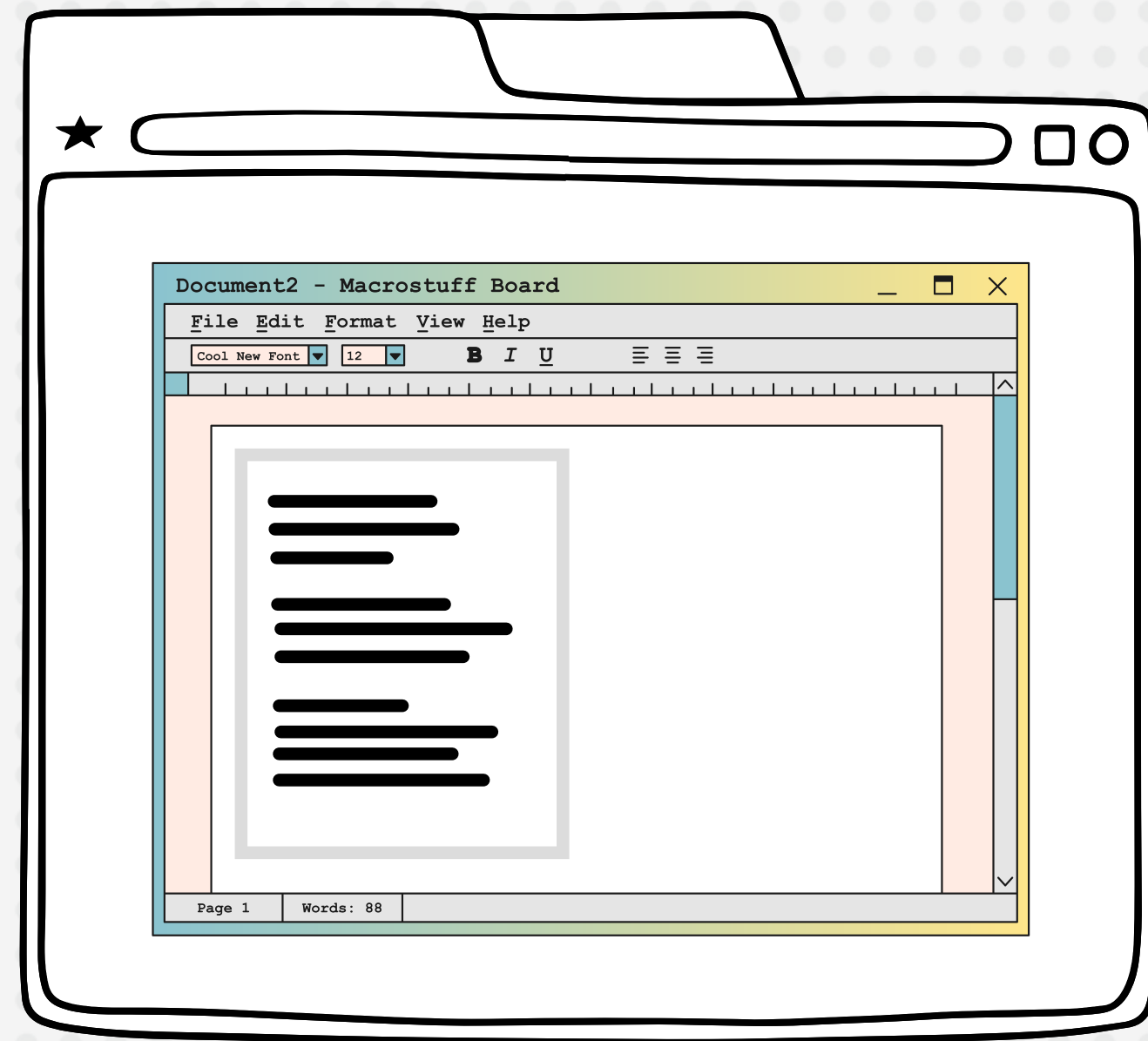




RESULTADOS

Criamos um código que dado um **arquivo de texto**, toda vez que o programa é executado criam-se **4 outros arquivos de back-up** do arquivo principal, um com letras, um com números, um com caracteres especiais e outro com o número de linhas usadas.

Usamos **5 threads**, 4 para os back-ups e um destinado ao controle de caracteres.



OBRIGADO



Repositório no GitHub:

https://github.com/Moab76/AnderssonSilva_AndrezaGon-alves_ws_OS_RR_2025

AnderssonSilva_AndrezaGon-alves_ws_OS_RR_2025 Public

Watch 0 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

File	Commit Message	Time
editor_texto	adicionando programa editor de texto com rust	2 days ago
README.md	Update README.md	yesterday
Slides_editor_texto.pdf	Add files via upload	5 hours ago
analise_pelo_tina.txt	Add files via upload	2 days ago
editor_texto_diagrama_de_petri.pn	Add files via upload	2 days ago

6 Commits

About

No description, website, or topics provided.

Readme

Activity

0 stars

0 watching

0 forks

Report repository

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors 2

Moab76 Andersson Silva

andrezaolive Andreza Gonçalves

Languages

Rust 89.2% Makefile 10.8%

Suggested workflows

Based on your tech stack

SLSA Generic generator

Generate SLSA3 provenance for your existing release workflows

Configure

AnderssonSilva_AndrezaGoncalves_ws_OS_RR_2025

Repositório de projetos desenvolvidos por Andersson Silva e Andreza Gonçalves para a disciplina de Sistemas Operacionais no curso de Ciência da Computação da Universidade Federal de Roraima (UFRR), no semestre 2025.1.

Estrutura do Projeto

Este repositório contém os seguintes diretórios:

- text_editor/ : Editor de texto desenvolvido em Rust, com funcionalidades básicas de edição e salvamento de arquivos.

Tecnologias Utilizadas

- Linguagem de Programação: [Rust](#)
- Controle de Versão: [Git](#)
- Plataforma de Hospedagem: [GitHub](#)

Como Executar os Projetos

Certifique-se de ter o Rust instalado em sua máquina. Caso não tenha, instale-o seguindo as instruções em: