MultiCoreScheduler

Este é um projeto de um escalonador para sistemas multicore implementado em Rust. O objetivo do projeto é gerenciar tarefas em um sistema com múltiplos núcleos de processamento, distribuindo a carga de trabalho de forma eficiente entre os núcleos.

Estrutura do Projeto

Codigos podem ser vistos na pastas SRC no link

`https://github.com/KaikeVitorino/SistemasOperacionais_4o/tree/main/MultiCo`

A estrutura de diretórios do projeto é a seguinte:

```
MultiCoreScheduler/
│
├── src/
│   ├── cpu.rs
│   ├── main.rs
│   ├── queue.rs
│   └── task.rs
│
├── target/
│   ├── debug/
│   ├── ...
│
├── .gitignore
├── Cargo.lock
└── Cargo.toml
```

Descrição dos Arquivos

- **src/cpu.rs**: Implementa a lógica relacionada à simulação de um núcleo de processamento (CPU).
- **src/main.rs**: Arquivo principal do projeto, responsável por iniciar o programa e orquestrar as chamadas às demais funções.
- **src/queue.rs**: Contém a implementação da fila de tarefas (queue) usada para armazenar e organizar as tarefas que serão escalonadas.
- **src/task.rs**: Define a estrutura e o comportamento das tarefas que serão gerenciadas pelo escalonador.
- **Cargo.toml**: Arquivo de configuração do Cargo, o gerenciador de pacotes do Rust, onde estão especificadas as dependências e as configurações do projeto.

- **Cargo.lock**: Arquivo gerado automaticamente que bloqueia as versões exatas das dependências do projeto.

Como Rodar o Projeto

Para compilar e rodar o projeto, siga os passos abaixo:

1. **Clone o repositório**:

   ```
   git clone https://github.com/KaikeVitorino/SistemasOperacionais_4o.git
   cd SistemasOperacionais_4o/MultiCoreScheduler
   ```

2. **Compile o projeto**:

   ```
   cargo build
   ```

   Isso irá compilar o código-fonte e gerar um executável na pasta `target/debug`.

3. **Execute o projeto**:

   ```
   cargo run
   ```

   Esse comando irá compilar (se necessário) e executar o programa.

Exemplo da saída do programa

```
Tempo: 0
Fila de processos: [Processo { nome: "Processo_A", tempo_chegada: 0, durac
Atribuindo processo ao processador 0: Processo { nome: "Processo_A", tempo
Atribuindo processo ao processador 1: Processo { nome: "Processo_B", tempo
Atribuindo processo ao processador 2: Processo { nome: "Processo_C", tempo
Atribuindo processo ao processador 3: Processo { nome: "Processo_D", tempo
Processador 0 está executando um processo, tempo restante: 4
Processador 1 está executando um processo, tempo restante: 2
Processador 2 está executando um processo, tempo restante: 6
Processador 3 está executando um processo, tempo restante: 0
Processo no processador 3 finalizado.
Finalizando processo: Processo_D
---
Tempo: 1
Fila de processos: [Processo { nome: "Processo_F", tempo_chegada: 6, durac
Atribuindo processo ao processador 3: Processo { nome: "Processo_F", tempo
Processador 0 está executando um processo, tempo restante: 3
Processador 1 está executando um processo, tempo restante: 1
Processador 2 está executando um processo, tempo restante: 5
Processador 3 está executando um processo, tempo restante: 3
---
Tempo: 2
Fila de processos: [Processo { nome: "Processo_E", tempo_chegada: 8, durac
Processador 0 está executando um processo, tempo restante: 2
```

```
Processador 1 está executando um processo, tempo restante: 0
Processo no processador 1 finalizado.
Finalizando processo: Processo_B
Processador 2 está executando um processo, tempo restante: 4
Processador 3 está executando um processo, tempo restante: 2
Tempo: 0
Fila de processos: [Processo { nome: "Processo_A", tempo_chegada: 0, durac
Atribuindo processo ao processador 0: Processo { nome: "Processo_A", tempo
Atribuindo processo ao processador 1: Processo { nome: "Processo_B", tempo
Atribuindo processo ao processador 2: Processo { nome: "Processo_C", tempo
Atribuindo processo ao processador 3: Processo { nome: "Processo_D", tempo
Processador 0 está executando um processo, tempo restante: 4
Processador 1 está executando um processo, tempo restante: 2
Processador 2 está executando um processo, tempo restante: 6
Processador 3 está executando um processo, tempo restante: 0
Processo no processador 3 finalizado.
Finalizando processo: Processo_D
---
Tempo: 1
Fila de processos: [Processo { nome: "Processo_F", tempo_chegada: 6, durac
Atribuindo processo ao processador 3: Processo { nome: "Processo_F", tempo
Processador 0 está executando um processo, tempo restante: 3
Processador 1 está executando um processo, tempo restante: 1
Processador 2 está executando um processo, tempo restante: 5
Processador 3 está executando um processo, tempo restante: 3
---
Tempo: 2
Fila de processos: [Processo { nome: "Processo_E", tempo_chegada: 8, durac
Processador 0 está executando um processo, tempo restante: 2
Processador 1 está executando um processo, tempo restante: 0
Processo no processador 1 finalizado.
Finalizando processo: Processo_B
Processador 2 está executando um processo, tempo restante: 4
Processador 3 está executando um processo, tempo restante: 2
---
Tempo: 3
Fila de processos: [Processo { nome: "Processo_E", tempo_chegada: 8, durac
Atribuindo processo ao processador 1: Processo { nome: "Processo_E", tempo
Processador 0 está executando um processo, tempo restante: 1
Processador 1 está executando um processo, tempo restante: 8
Processador 2 está executando um processo, tempo restante: 3
Processador 3 está executando um processo, tempo restante: 1
---
```

Dependências

O projeto não possui dependências externas além do próprio Rust, que é gerenciado pelo Cargo. Certifique-se de ter o Rust instalado na sua máquina. Você pode instalar o Rust usando o comando:

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```