Simulador de Escalonamento de Tarefas

JOÃO DANIEL MEIRA FARIA

Apresentação

- O código elaborado implementa um simulador para escalonamento de tarefas em tempo real, usando os algoritmos Rate Monotonic (RM) e Earliest Deadline First (EDF) apresentando-os em um Gráfico de Gantt.
- Foi utilizado a linguagem de programação Python versão 3.12.5
- Para a exibição do Gráfico de Gantt é necessário a instalação e importação da lib `matplotlib`
- `import matplotlib.pyplot as plt`

Escalonabilidade

- Foram criadas as seguintes funções para definir se o conjunto de tarefas é escalonável:
- `def escalonabilidade_rm(tarefas):`
- `def escalonabilidade_edf(tarefas):`

escalonabilidade_rm(tarefas):


```
def escalonabilidade_rm(tarefas):
    """
    Calcula a escalonabilidade usando a condição de Liu & Layland para RM.
    :param tarefas: Lista de tuplas (tempo_execucao, periodo)
    :return: Tupla (escalonavel, utilizacao_total)
    """
    n = len(tarefas)
    utilizacao_total = sum(t[0] / t[1] for t in tarefas)
    limite_rm = n * (2**(1/n) - 1)  # Limite de utilização para RM
    return utilizacao_total <= limite_rm, utilizacao_total
```

escalonabilidade_edf(tarefas)


```
def escalonabilidade_edf(tarefas):
    """
    Calcula a escalonabilidade para EDF.
    :param tarefas: Lista de tuplas (tempo_execucao, periodo)
    :return: Tupla (escalonavel, utilizacao_total)
    """
    utilizacao_total = sum(t[0] / t[1] for t in tarefas)
    return utilizacao_total <= 1, utilizacao_total  # EDF é escalonável se utilização <= 100%
```

Gráfico de Gantt


```
def gerar_grafico_gantt(tarefas, tipo_escalonamento):
    """
    Gera o gráfico de Gantt para o escalonamento das tarefas.

    :param tarefas: Lista de tuplas (tempo_execucao, periodo)
    :param tipo_escalonamento: String 'RM' ou 'EDF'
    """
    fig, gantt = plt.subplots()

    # Configuração inicial
    task_names = [f'Tarefa {i+1}' for i in range(len(tarefas))]
    cores = ['tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'tab:purple', 'tab:brown']

    tempo_total = max(t[1] for t in tarefas) * 3  # Simula por 3 vezes o maior período
```

```
# Inicialização de estruturas de dados para o escalonamento
    exec_times = {i: 0 for i in range(len(tarefas))}
    periodos = {i: tarefas[i][1] for i in range(len(tarefas))}
    proximos_deadlines = {i: periodos[i] for i in range(len(tarefas))}

    tempo_atual = 0
    eventos = []
```

```
    # Desenhar o gráfico de Gantt
        for inicio, duracao, tarefa in eventos:
            gantt.broken_barh([(inicio, duracao)], (tarefa * 10, 9), facecolors=(cores[tarefa % len(cores)]))

        # Adicionar linhas verticais para marcar os intervalos de tempo
        for t in range(tempo_total + 1):
            gantt.axvline(t, color='gray', linestyle='--', linewidth=0.5)

        # Configurar os eixos e labels do gráfico
        gantt.set_ylim(0, len(tarefas) * 10)
        gantt.set_xlim(0, tempo_total)
        gantt.set_xlabel('Tempo')
        gantt.set_yticks([i * 10 + 5 for i in range(len(tarefas))])
        gantt.set_yticklabels(task_names)
        gantt.set_xticks(range(0, tempo_total + 1, 1))

        plt.title(f'Escalonamento {tipo_escalonamento}')
        plt.show()
```

Escalonador


```
def escalonar(tarefas, tipo_escalonamento, tempo_atual):
        """Seleciona a próxima tarefa a ser executada baseado no tipo de escalonamento."""
        if tipo_escalonamento == 'RM':
            return min(tarefas, key=lambda x: periodos[x])  # RM: menor período primeiro
        elif tipo_escalonamento == 'EDF':
            return min(tarefas, key=lambda x: proximos_deadlines[x])  # EDF: menor deadline primeiro

    # Loop principal de simulação
    while tempo_atual < tempo_total:
        # Atualizar tempos de execução e deadlines no início de cada período
        for i in range(len(tarefas)):
            if tempo_atual % periodos[i] == 0:
                exec_times[i] = tarefas[i][0]
                proximos_deadlines[i] = tempo_atual + periodos[i]
```

```
    # Identificar tarefas prontas para execução
        tarefas_prontas = [i for i in range(len(tarefas)) if exec_times[i] > 0]

        if tarefas_prontas:
            # Selecionar e executar a próxima tarefa
            tarefa_atual = escalonar(tarefas_prontas, tipo_escalonamento,
tempo_atual)
            eventos.append((tempo_atual, 1, tarefa_atual))
            exec_times[tarefa_atual] -= 1
            tempo_atual += 1
        else:
            # Avançar o tempo se não houver tarefas prontas
            tempo_atual += 1
```

Interação com Usuário


```
def simulador_escalonador():
    """Função principal que interage com o usuário e coordena a simulação."""
    print("Bem-vindo ao Simulador de Escalonamento!")
    print("Escolha uma opção:")
    print("1. Usar conjunto de teste predefinido")
    print("2. Inserir tarefas manualmente")

    escolha = input("Sua escolha (1 ou 2): ")

    if escolha == '1':
        conjuntos_teste = obter_conjunto_teste()
        print("\nConjuntos de teste disponíveis:")
        for i, nome in enumerate(conjuntos_teste.keys(), 1):
            print(f"{i}. {nome}")

```

```

escolha_conjunto = int(input("\nEscolha o número do conjunto de teste: ")) - 1
        tarefas = list(conjuntos_teste.values())[escolha_conjunto]
        print(f"\nConjunto selecionado: {list(conjuntos_teste.keys())[escolha_conjunto]}")
        print("Tarefas:", tarefas)
    elif escolha == '2':
        # Coletar dados das tarefas manualmente
        n = int(input("Digite o número de tarefas: "))
        tarefas = []
        for i in range(n):
            exec_time = int(input(f"Digite o tempo de execução da tarefa {i+1}: "))
            period = int(input(f"Digite o período da tarefa {i+1}: "))
            tarefas.append((exec_time, period))
    else:
        print("Escolha inválida. Encerrando o programa.")
        return
```

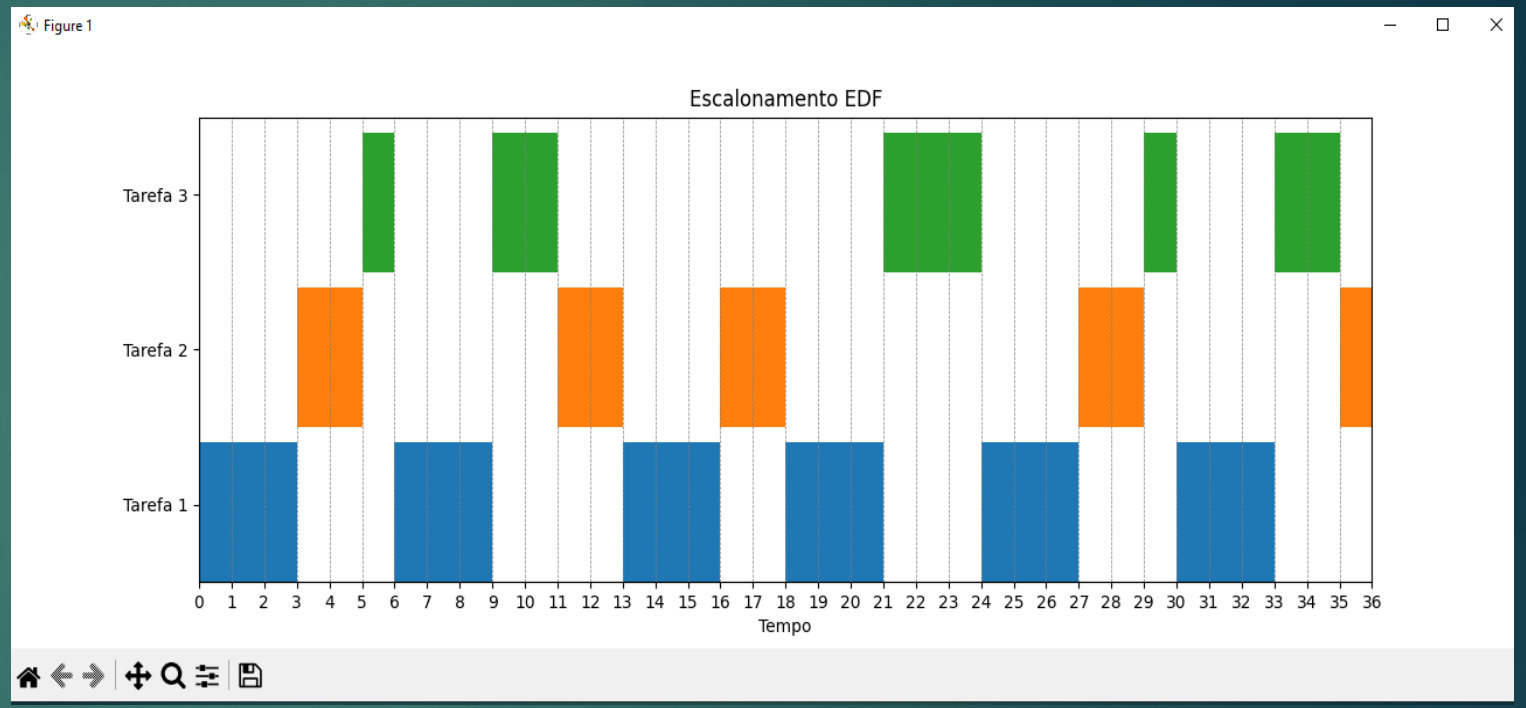
```
    # Selecionar o tipo de escalonamento
    tipo_escalonamento = input("Escolha o tipo de escalonamento (RM/EDF): ").upper()

    # Verificar a escalonabilidade e gerar o gráfico se for escalonável
    if tipo_escalonamento == 'RM':
        escalonavel, utilizacao = escalonabilidade_rm(tarefas)
    elif tipo_escalonamento == 'EDF':
        escalonavel, utilizacao = escalonabilidade_edf(tarefas)
    else:
        print("Tipo de escalonamento inválido.")
        return

    print(f"\nUtilização total: {utilizacao:.2f}")
    if escalonavel:
        print("As tarefas são escalonáveis.")
        gerar_grafico_gantt(tarefas, tipo_escalonamento)
    else:
        print("As tarefas NÃO são escalonáveis. Nenhum gráfico de Gantt será gerado.")

# Ponto de entrada do programa
if __name__ == "__main__":
    simulador_escalonador()
```

Conjunto de Testes


```
def obter_conjunto_teste():
    """
    Fornece conjuntos de teste predefinidos para o simulador.

    :return: Um dicionário com conjuntos de teste nomeados
    """
    return {
        "Conjunto 1 (RM Escalonável)": [
            (1, 8),  # Tarefa 1: Execução = 1, Período = 8
            (2, 12), # Tarefa 2: Execução = 2, Período = 12
            (3, 24)  # Tarefa 3: Execução = 3, Período = 24
        ],
        "Conjunto 2 (RM Não Escalonável)": [
            (3, 8),  # Tarefa 1: Execução = 3, Período = 8
            (4, 12), # Tarefa 2: Execução = 4, Período = 12
            (5, 24)  # Tarefa 3: Execução = 5, Período = 24
        ],
        "Conjunto 3 (EDF Escalonável)": [
            (3, 6),  # Tarefa 1: Execução = 3, Período = 6
            (2, 8),  # Tarefa 2: Execução = 2, Período = 8
            (3, 12)  # Tarefa 3: Execução = 3, Período = 12
        ],
        "Conjunto 4 (EDF Não Escalonável)": [
            (4, 6),  # Tarefa 1: Execução = 4, Período = 6
            (3, 8),  # Tarefa 2: Execução = 3, Período = 8
            (4, 12)  # Tarefa 3: Execução = 4, Período = 12
        ],
        "Conjunto 5 (Misto)": [
            (1, 4),  # Tarefa 1: Execução = 1, Período = 4
            (2, 6),  # Tarefa 2: Execução = 2, Período = 6
            (3, 8),  # Tarefa 3: Execução = 3, Período = 8
            (2, 12)  # Tarefa 4: Execução = 2, Período = 12
        ]
    }
```

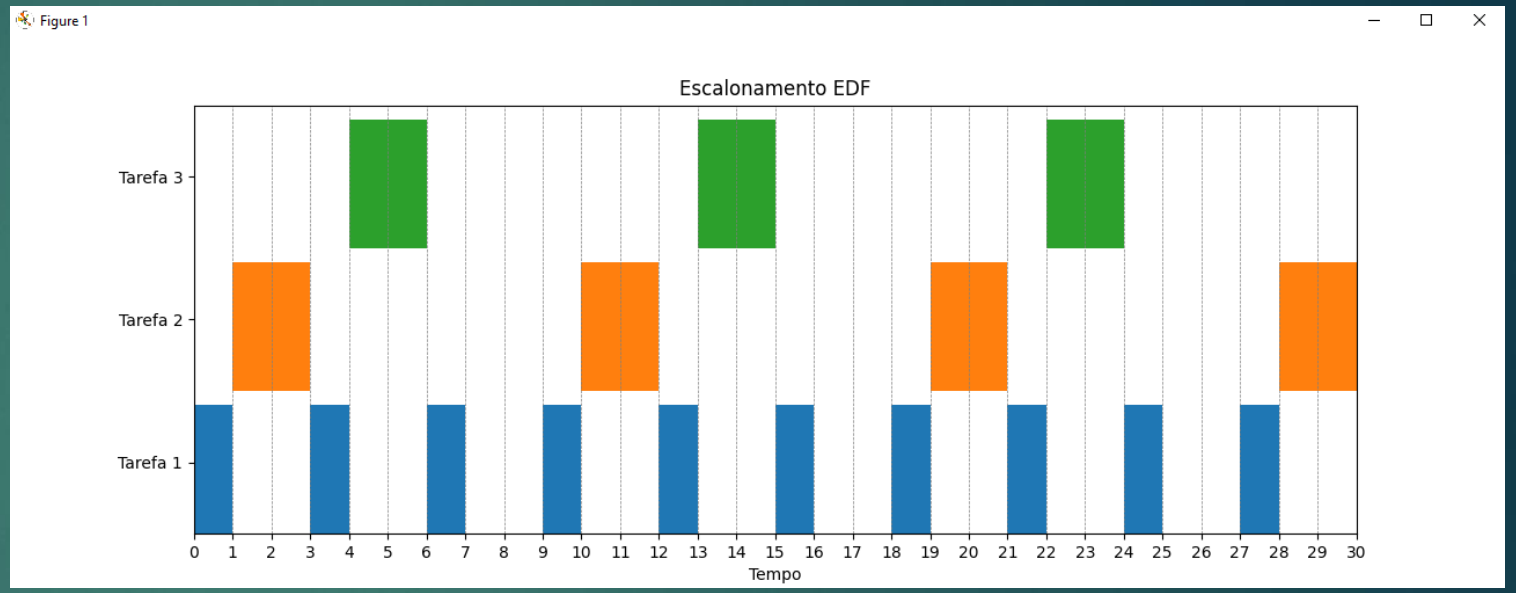
Execução


```
Prompt de Comando - python  main.py

C:\Users\JDaniel\Desktop\STR>python main.py
Bem-vindo ao Simulador de Escalonamento!
Escolha uma opção:
1. Usar conjunto de teste predefinido
2. Inserir tarefas manualmente
Sua escolha (1 ou 2): 1

Conjuntos de teste disponíveis:
1. Conjunto 1 (RM Escalonável)
2. Conjunto 2 (RM Não Escalonável)
3. Conjunto 3 (EDF Escalonável)
4. Conjunto 4 (EDF Não Escalonável)
5. Conjunto 5 (Misto)

Escolha o número do conjunto de teste: 3

Conjunto selecionado: Conjunto 3 (EDF Escalonável)
Tarefas: [(3, 6), (2, 8), (3, 12)]
Escolha o tipo de escalonamento (RM/EDF): EDF
```



Prompt de Comando - python main.py

```
C:\Users\JDaniel\Desktop\STR>python main.py
Bem-vindo ao Simulador de Escalonamento!
Escolha uma opção:
1. Usar conjunto de teste predefinido
2. Inserir tarefas manualmente
Sua escolha (1 ou 2): 2
Digite o número de tarefas: 3
Digite o tempo de execução da tarefa 1: 1
Digite o período da tarefa 1: 3
Digite o tempo de execução da tarefa 2: 2
Digite o período da tarefa 2: 9
Digite o tempo de execução da tarefa 3: 2
Digite o período da tarefa 3: 10
Escolha o tipo de escalonamento (RM/EDF): EDF

Utilização total: 0.76
As tarefas são escalonáveis.
```



```
C:\Users\JDaniel\Desktop\STR>python main.py
Bem-vindo ao Simulador de Escalonamento!
Escolha uma opção:
1. Usar conjunto de teste predefinido
2. Inserir tarefas manualmente
Sua escolha (1 ou 2): 2
Digite o número de tarefas: 4
Digite o tempo de execução da tarefa 1: 1
Digite o período da tarefa 1: 2
Digite o tempo de execução da tarefa 2: 2
Digite o período da tarefa 2: 6
Digite o tempo de execução da tarefa 3: 2
Digite o período da tarefa 3: 13
Digite o tempo de execução da tarefa 4: 1
Digite o período da tarefa 4: 15
Escolha o tipo de escalonamento (RM/EDF): RM

Utilização total: 1.05
As tarefas NÃO são escalonáveis. Nenhum gráfico de Gantt será gerado.
```

Github

- https://github.com/jdanielmf/RTS

Referências

J.-M. Farines, J. da S. Fraga, R. S. de Oliveira. Sistemas de Tempo Real. Escola de Computação 2000, IME-USP, São Paulo-SP, julho/2000.

Jane Liu. Real-Time Systems. Prentice-Hall, 2000.

A. Burns, A. Wellings. Real-Time Systems and Programming Languages. 3nd ed. Addison-Wesley, 2005