Sistemas Operacionais

Curso: Engenharia da computação

Prof.: Maurício Acconcia Dias

Data: 01/12/2025

Simulador Sistema Operacional

Implementação de entrada e saída

Caroline da Silva Grizante Ra:114105

Emilly Emanuelly R. dos Santos Ra:114095

Marcela Lovatto Ra:113626

Fundação Hermínio Ometto

Araras-SP

Sumário

1. Introdução

Este relatório detalhado apresenta a análise da estrutura de um Simulador de Sistema Operacional (SO), desenvolvido em C#. O projeto visa replicar e simular os principais componentes e funcionalidades de um SO moderno, proporcionando um ambiente controlado para estudo e experimentação de conceitos fundamentais de sistemas operacionais, como gerenciamento de processos, escalonamento de CPU, gerenciamento de memória e sistema de arquivos. A arquitetura do simulador é modular, organizada em dez pastas principais, cada uma dedicada a um subsistema específico do SO.

2. Objetivo do Projeto

O objetivo principal deste projeto é criar uma ferramenta de simulação completa e funcional que permita a visualização e o teste de diferentes algoritmos e políticas de gerenciamento de recursos de um Sistema Operacional. Especificamente, o simulador busca:

- Implementar um núcleo central que coordene todas as operações.
- Simular o ciclo de vida de processos e threads.
- Permitir a comparação de diversos algoritmos de escalonamento de CPU.
- Modelar o gerenciamento de memória com paginação e o uso de estruturas como a TLB.
- Simular operações de Entrada/Saída e um Sistema de Arquivos básico.
- Fornecer métricas detalhadas para avaliar o desempenho dos subsistemas simulados.

3. Desenvolvimento do Sistema

O desenvolvimento do simulador é estruturado em dez módulos, cada um contendo classes específicas que implementam as funcionalidades do subsistema correspondente. A seguir, detalhamos a função de cada classe por módulo.

3.1. Módulo Núcleo (Nucleo/)

Este módulo contém os componentes centrais que orquestram o funcionamento do simulador.

Classe	Função
Kernel.cs	Núcleo Central: Atua como o ponto de integração e controle de todos os outros componentes do simulador, sendo o coração do sistema.
Relogio.cs	Controle de Tempo: Simula o relógio do sistema, essencial para o controle de tempo, timeouts e a execução sequencial de eventos.
RegistradorDeEventos.cs	Sistema de Logging: Responsável por registrar e armazenar todos os eventos e operações importantes que ocorrem durante a simulação, facilitando a depuração e a análise.
Configuracoes.cs	Configurações Globais: Armazena e gerencia parâmetros globais que definem o comportamento da simulação (ex: tamanho da memória, tempo de quantum).
CarregadorWorkload.cs	Carregamento de Carga de Trabalho: Lê e interpreta arquivos de workload (carga de trabalho) que definem os processos e as tarefas a serem executadas na simulação.

3.2. Módulo Gerenciamento de Processos (Processos/)

Responsável pela criação, controle e término dos processos.

Classe	Função
Processo.cs	Representação do Processo: Define a estrutura de dados e o comportamento de um processo individual no simulador.
PCB.cs	Process Control Block: Armazena todas as informações de estado e contexto necessárias para o gerenciamento de um processo pelo núcleo (ex: registradores, estado, ponteiros de memória).
EstadoProcesso.cs	Enumeração de Estados: Define os estados possíveis em que um processo pode se encontrar (ex: Novo, Pronto, Executando, Espera, Terminado).

GerenciadorDeProcessos.cs	Gerenciamento Central: Controla o ciclo de vida dos processos, incluindo criação, suspensão, retomada e término.

3.3. Módulo Gerenciamento de Threads (Threads/)

Estende o gerenciamento de processos para incluir a simulação de threads dentro de um processo.

Classe	Função
ThreadSimulada.cs	Representação da Thread: Define a estrutura de dados e o comportamento de uma thread individual.
TCB.cs	Thread Control Block: Armazena as informações de estado e contexto específicas de uma thread.
EstadoThread.cs	Enumeração de Estados: Define os estados possíveis em que uma thread pode se encontrar.
GerenciadorDeThreads.cs	Gerenciamento Central: Controla a criação, execução e sincronização das threads dentro dos processos.

3.4. Módulo Escalonamento (Escalonamento/)

Implementa os algoritmos e a lógica para decidir qual processo ou thread deve ser executado pela CPU.

Classe	Função
IAlgoritmoEscalonamento.cs	Interface de Algoritmos: Define o contrato que todos os algoritmos de escalonamento devem seguir, permitindo a fácil adição de novos algoritmos.
FCFS.cs	First-Come, First-Served: Implementa o algoritmo de escalonamento não preemptivo mais simples.
RoundRobin.cs	Round Robin: Implementa o algoritmo preemptivo baseado em quantum de tempo.

PrioridadePreemptivo.cs	Prioridade Preemptivo: Implementa o escalonamento que permite a interrupção de um processo de menor prioridade por um de maior prioridade.
PrioridadeNaoPreemptivo.cs	Prioridade Não Preemptivo: Implementa o escalonamento onde um processo de maior prioridade só é executado após o término do processo atual.
FilaProntos.cs	Fila de Processos Prontos: Estrutura de dados que armazena os processos ou threads que estão prontos para serem executados.
Escalonador.cs	Escalonador Principal: Componente que utiliza a FilaProntos e o algoritmo selecionado para tomar a decisão de escalonamento.
TrocaDeContexto.cs	Controle de Contexto: Simula o processo de salvar o estado de um processo em execução e carregar o estado de outro processo (o overhead da troca de contexto).

3.5. Módulo Memória (Memoria/)

Dedica-se à simulação do gerenciamento de memória principal, incluindo paginação e memória virtual.

Classe	Função
GerenciadorDeMemoria.cs	Gerenciamento Central: Controla a alocação e desalocação de memória física e virtual.
Pagina.cs	Representação da Página: Define a unidade de memória virtual de um processo.
TabelaDePaginas.cs	Mapeamento Virtual-Físico: Armazena o mapeamento entre páginas virtuais e molduras físicas para um processo.
Moldura.cs	Representação da Moldura (Frame): Define a unidade de memória física.

TabelaDeMolduras.cs	Controle de Memória Física: Mantém o registro do estado de todas as molduras de memória física.
PoliticaAlocacao.cs	Estratégias de Alocação: Implementa diferentes políticas para alocar espaço na memória (ex: First-Fit, Best-Fit).
EntradaTLB.cs	Entrada do TLB: Define a estrutura de uma entrada no Translation Lookaside Buffer.
TLB.cs	Translation Lookaside Buffer: Simula o cache de mapeamento de endereços, acelerando a tradução de endereços virtuais para físicos.

3.6. Módulo Entrada/Saída (EntradaSaida/)

Simula a interação do sistema com dispositivos de Entrada/Saída.

Classe	Função
IDispositivo.cs	Interface de Dispositivos: Define o contrato básico para todos os dispositivos de E/S simulados.
DispositivoDeBloco.cs	Dispositivos de Bloco: Simula dispositivos que transferem dados em blocos (ex: disco rígido).
DispositivoDeCaractere.cs	Dispositivos de Caractere: Simula dispositivos que transferem dados caractere por caractere (ex: teclado, impressora).
RequisicaoES.cs	Requisição de E/S: Representa uma solicitação de um processo para realizar uma operação de E/S.
GerenciadorES.cs	Gerenciamento Central: Controla o fluxo de requisições de E/S, enfileiramento e atendimento pelos dispositivos.
Interrupcao.cs	Sistema de Interrupções: Simula o mecanismo pelo qual os dispositivos notificam a CPU sobre a conclusão de uma operação de E/S.

3.7. Módulo Sistema de Arquivos (SistemaDeArquivos/)

Simula a organização e o gerenciamento de dados persistentes.

Classe	Função
SistemaDeArquivos.cs	Sistema de Arquivos Principal: Componente central que gerencia a estrutura lógica e física dos arquivos.
FCB.cs	File Control Block: Estrutura de dados que armazena metadados sobre um arquivo (ex: tamanho, permissões, localização dos blocos).
EntradaArquivo.cs	Representação de Arquivo: Define a estrutura de um arquivo dentro do sistema.
EntradaDiretorio.cs	Representação de Diretório: Define a estrutura de um diretório, contendo referências a arquivos e outros diretórios.
ManipuladorArquivo.cs	Manipulação de Arquivos: Implementa as operações básicas de arquivo (ex: abrir, ler, escrever, fechar).
TabelaDeAlocacao.cs	Tabela de Alocação de Blocos: Simula a estrutura de dados que rastreia quais blocos de disco estão livres e quais estão ocupados por arquivos (ex: FAT, i-nodes).

3.8. Módulo Métricas (Metricas/)

Fornece a capacidade de coletar e analisar dados de desempenho da simulação.

Classe	Função
GerenciadorDeMetricas.cs	Gerenciamento Central: Coleta, armazena e processa os dados de desempenho de todos os subsistemas.
MetricasProcesso.cs	Métricas por Processo: Registra dados como tempo de espera, tempo de resposta e throughput por processo.
MetricasDispositivo.cs	Métricas de Dispositivos: Registra dados de utilização e tempo de serviço dos dispositivos de E/S.

MetricasMemoria.cs	Métricas de Memória: Registra dados como taxa de acerto/erro da TLB, taxa de falta de página e utilização da memória.

3.9. Módulo Interface (Interface/)

Responsável pela interação do usuário com o simulador, provavelmente via console ou terminal.

Classe	Função
MenuPrincipal.cs	Menu Inicial: Ponto de entrada para a interface do usuário, direcionando para os submenus.
MenuProcessos.cs	Menu de Processos: Permite ao usuário interagir com o gerenciador de processos (ex: criar, listar, matar processos).
MenuThreads.cs	Menu de Threads: Permite ao usuário interagir com o gerenciador de threads.
MenuEscalonamento.cs	Menu de Escalonamento: Permite ao usuário selecionar o algoritmo de escalonamento a ser usado e visualizar as filas.
MenuMemoria.cs	Menu de Memória: Permite ao usuário visualizar o estado da memória, tabelas de páginas e TLB.
MenuES.cs	Menu de E/S: Permite ao usuário visualizar o estado dos dispositivos e as requisições pendentes.
MenuArquivo.cs	Menu de Arquivos: Permite ao usuário interagir com o sistema de arquivos (ex: criar, ler, listar arquivos).
MenuMetricas.cs	Menu de Métricas: Permite ao usuário visualizar os relatórios de desempenho gerados.
MenuConfiguracoes.cs	Menu de Configurações: Permite ao usuário ajustar os parâmetros globais da simulação.

3.10. Módulo Utilitários (Utilitarios/)

Contém classes de suporte para funcionalidades gerais do sistema.

Classe	Função
GeradorIDs.cs	Geração de Identificadores: Fornece um mecanismo para gerar IDs únicos para processos, threads e outros objetos do sistema.
GeradorAleatorio.cs	Geração de Aleatoriedade: Fornece números aleatórios para simular eventos não determinísticos (ex: tempo de E/S, chegada de processos).

4. Aplicação e Relevância

O Simulador de Sistema Operacional em C# possui uma relevância significativa, principalmente no contexto educacional e de pesquisa.

- Ferramenta Educacional: Serve como um laboratório virtual para estudantes de Ciência da Computação e Engenharia, permitindo que eles visualizem e compreendam a complexidade e a interação dos componentes de um SO. A capacidade de trocar algoritmos de escalonamento ou políticas de alocação de memória e observar o impacto nas métricas é inestimável.
- Pesquisa e Desenvolvimento: O projeto pode ser utilizado para testar e validar novos algoritmos de gerenciamento de recursos em um ambiente controlado, antes de uma implementação em um sistema real.
- Análise de Desempenho: O módulo de Métricas permite uma análise quantitativa do desempenho do sistema sob diferentes cargas de trabalho (workloads), fornecendo insights sobre a eficiência das políticas implementadas.

5. Conclusão

O projeto do Simulador de Sistema Operacional em C# demonstra uma arquitetura robusta e abrangente, cobrindo todos os subsistemas cruciais de um SO. A organização modular, com classes bem definidas para cada conceito (PCB, TCB, TLB, FCB, etc.), indica um design de alta qualidade e aderência aos princípios teóricos de sistemas operacionais. A inclusão de múltiplos algoritmos de escalonamento e políticas de alocação, juntamente com

um sistema de métricas detalhadas, garante que o simulador seja uma ferramenta poderosa para o estudo aprofundado e a experimentação prática.

6. Imagens

1- Como ficou estruturada as páginas

```
SimuladorSO_Final/
├── SimuladorSOLogica/          # Biblioteca com toda a lógica do simulador
│   ├── Nucleo/                 # Kernel, Relógio, Registrador, Configurações
│   ├── Processos/              # Gerenciamento de processos
│   ├── Threads/                # Gerenciamento de threads
│   ├── Escalonamento/          # Algoritmos de escalonamento
│   ├── Memoria/                # Gerenciamento de memória
│   ├── EntradaSaida/           # Sistema de E/S
│   ├── SistemaDeArquivos/      # Sistema de arquivos
│   ├── Metricas/               # Coleta de métricas
│   ├── Interface/              # Menus console
│   └── Utilitarios/            # Utilitários gerais
│
├── SimuladorSOInterface/       # Interface WPF com tema escuro
│   ├── MainWindow.xaml         # Janela principal
│   ├── RelatorioWindow.xaml    # Gerenciamento de processos
│
└── workload_exemplo.txt        # Arquivo de exemplo de workload
```

2- Arquivo TXT

```
# ============================
# WORKLOAD PARA SIMULADOR DE SO
# ============================

# ---- Configurações do início da simulação ---
SET_SEED 1234
SET_QUANTUM 4
SET_ESCALONADOR RR
SET_TAMANHO_PAGINA 1024
SET_FRAMES 16

# =======================
# 1. Processos
# =======================

# Criar processos (PID simbólico + prioridade)
CRIAR_PROCESSO P1 3
CRIAR_PROCESSO P2 1
```

```
CRIAR_THREAD P2
CRIAR_THREAD P3
CRIAR_THREAD P4
CRIAR_THREAD P4
CRIAR_THREAD P5
CRIAR_THREAD P6
CRIAR_THREAD P7

# ======================
# 3. Memória
# ======================

# Solicitar alocação de memória (em bytes)
MEM_ALOCAR P1 4096
MEM_ALOCAR P2 2048
MEM_ALOCAR P3 8192
MEM_ALOCAR P4 1024
MEM_ALOCAR P5 4096
MEM_ALOCAR P6 2048
MEM_ALOCAR P7 8192

# Simular acessos de página
MEM_ACESSO P1 0x0003
MEM_ACESSO P1 0x0040
MEM_ACESSO P2 0x000A
MEM_ACESSO P3 0x0100
MEM_ACESSO P4 0x0001
MEM_ACESSO P5 0x0005
MEM_ACESSO P6 0x000B
MEM_ACESSO P7 0x0101
```

```
# ======================
# 4. CPU / Escalonamento
# ======================

# Processar ticks de CPU
CPU_TICK 10
CPU_TICK 10

# Mudar escalonador em tempo de execução (opcional)
SET_ESCALONADOR PRIORIDADE_PREEMPTIVO

CPU_TICK 10

# ======================
# 5. Entrada / Saída
# ======================
```

3- Aplicação pelo Console

```
[T=0] Dispositivo criado: DISCO (Tipo: bloco)
[T=0] Dispositivo criado: TECLADO (Tipo: caractere)
[T=0] Dispositivo criado: IMPRESSORA (Tipo: bloco)

============== SISTEMA OPERACIONAL - SIMULADOR ==============
1) Gerenciar Processos
2) Gerenciar Threads
3) Escalonador de CPU
4) Gerenciamento de Memória
5) Entrada e Saída (I/O)
6) Sistema de Arquivos
7) Estatísticas e Métricas
8) Configurações do Simulador
0) Sair
============================================================
Escolha uma opção: 8

------------------ CONFIGURAÇÕES ----------------------
1) Definir semente determinística
2) Configurar tamanho de página
3) Configurar número de molduras
4) Configurar tempos de dispositivos
5) Carregar workload
0) Voltar
-------------------------------------------------------
Escolha uma opção: 5

Caminho do arquivo de workload: workload_exemplo.txt

=== Carregando workload: workload_exemplo.txt ===
```

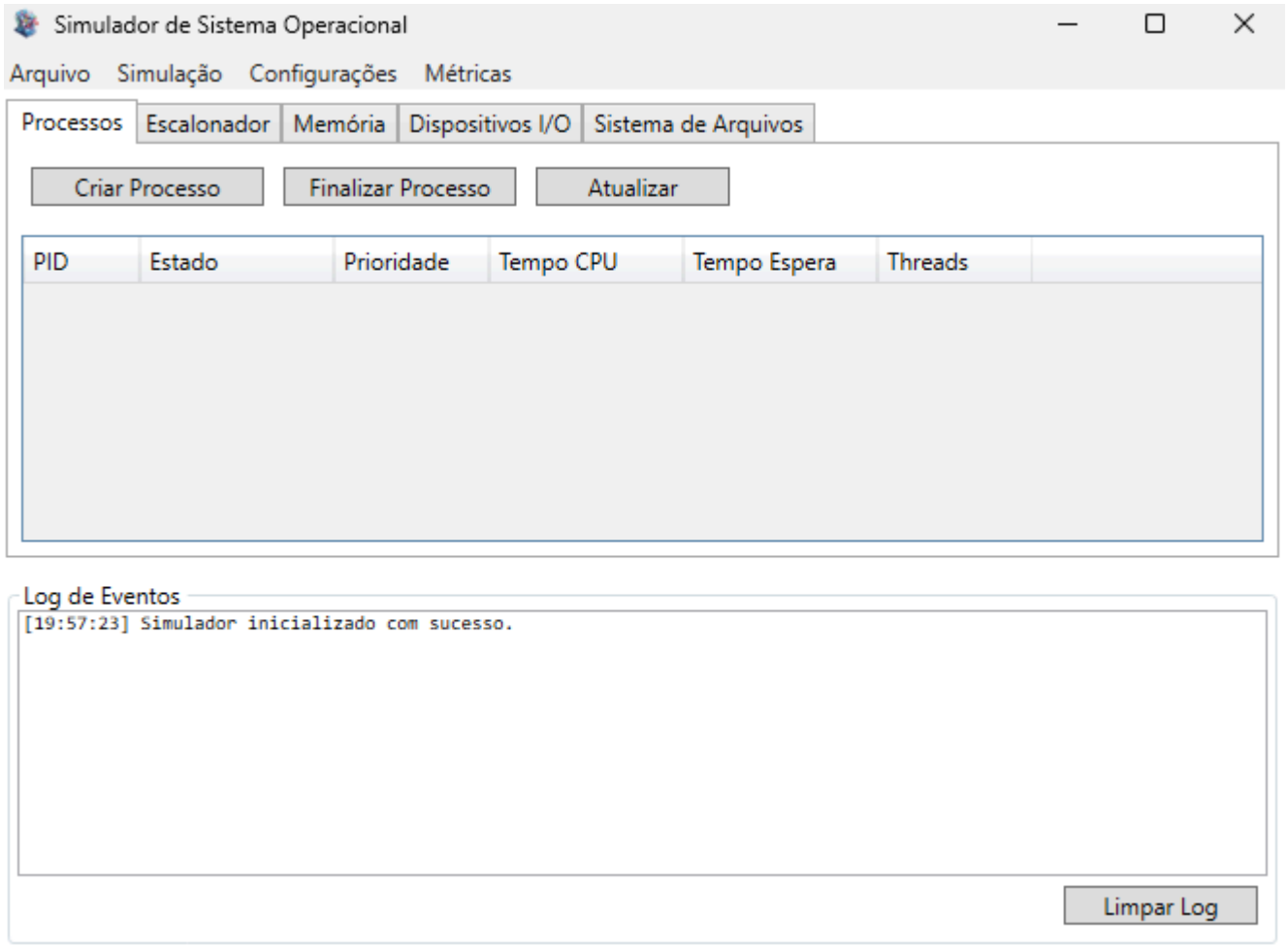
```
[T=0] Algoritmo de escalonamento alterado para: Round
[T=0] Processo criado: P1 (PID=1, Prioridade=3)
[T=0] Processo criado: P2 (PID=2, Prioridade=1)
[T=0] Processo criado: P3 (PID=3, Prioridade=2)
[T=0] Processo criado: P4 (PID=4, Prioridade=4)
[T=0] Processo criado: P5 (PID=5, Prioridade=1)
[T=0] Processo criado: P6 (PID=6, Prioridade=3)
[T=0] Processo criado: P7 (PID=7, Prioridade=2)
[T=0] Thread criada: TID=1 no processo P1 (PID=1)
[T=0] Thread criada: TID=2 no processo P1 (PID=1)
[T=0] Thread criada: TID=3 no processo P2 (PID=2)
[T=0] Thread criada: TID=4 no processo P3 (PID=3)
[T=0] Thread criada: TID=5 no processo P4 (PID=4)
```

```
╔═══════════════════════════════════════╗
║      RELATÓRIO COMPLETO DE MÉTRICAS   ║
╚═══════════════════════════════════════╝


===== TEMPO DE RETORNO POR PROCESSO =====
P1: 60 ticks
P2: 60 ticks
P3: 60 ticks
P4: 60 ticks
P5: 60 ticks
P6: 60 ticks
P7: 60 ticks

Média: 60,00 ticks
========================================
```

3- Aplicação com Interface Visual



PID	Estado	Prioridade	Tempo CPU	Tempo Espera	Threads
0	Pronto	3	4	26	2
1	Pronto	1	8	23	1
2	Executando	2	10	9	1
3	Pronto	4	0	10	2
4	Pronto	1	8	22	1
5	Pronto	3	0	0	1
6	Pronto	2	0	0	1
7	Finalizado	3	0	0	2
8	Finalizado	1	0	0	1
9	Finalizado	2	0	0	1
10	Finalizado	4	0	0	2
11	Finalizado	1	0	0	1
12	Finalizado	3	0	0	1
13	Finalizado	2	0	0	1

Algoritmo: Prioridade (Preemptivo)

Processo Atual: PID 2

Trocas de Contexto: 6

Fila de Prontos

```
PID: 0, Estado: Pronto, Prioridade: 3, Tempo CPU: 4, Threads: 2
PID: 1, Estado: Pronto, Prioridade: 1, Tempo CPU: 8, Threads: 1
PID: 4, Estado: Pronto, Prioridade: 1, Tempo CPU: 8, Threads: 1
PID: 3, Estado: Pronto, Prioridade: 4, Tempo CPU: 0, Threads: 2
PID: 6, Estado: Pronto, Prioridade: 2, Tempo CPU: 0, Threads: 1
PID: 8, Estado: Finalizado, Prioridade: 1, Tempo CPU: 0, Threads: 1
PID: 11, Estado: Finalizado, Prioridade: 1, Tempo CPU: 0, Threads: 1
PID: 5, Estado: Pronto, Prioridade: 3, Tempo CPU: 0, Threads: 1
```

Molduras: 16, Tamanho Página: 1024 bytes

Faltas de Página: 14

TLB: Desativada

Moldura	Ocupada	PID	Página
0	True	7	0
1	True	8	0
2	True	9	0
3	True	10	0
4	True	11	0
5	True	12	0
6	True	13	0
7	False	-1	-1

Dispositivos

Nome	Tipo	Ocupado	Tempo Op.
DISCO	Bloco	True	30
TECLADO	Caractere	False	10
IMPRESSORA	Caractere	True	40
REDE	Bloco	False	20

Interrupções

```
Interrupção 7: Dispositivo=TECLADO, PID=1, Tempo=30, Mensagem=Operação de I/O concluída, P
Interrupção 8: Dispositivo=TECLADO, PID=4, Tempo=30, Mensagem=Operação de I/O concluída, P
Interrupção 9: Dispositivo=DISCO, PID=0, Tempo=30, Mensagem=Operação de I/O concluída, Prc
Interrupção 10: Dispositivo=IMPRESSORA, PID=2, Tempo=40, Mensagem=Operação de I/O concluíc
Interrupção 11: Dispositivo=DISCO, PID=3, Tempo=50, Mensagem=Operação de I/O concluída, Pr
Interrupção 19: Dispositivo=DISCO, PID=6, Tempo=60, Mensagem=Operação de I/O concluída, Pr
Interrupção 20: Dispositivo=TECLADO, PID=8, Tempo=60, Mensagem=Operação de I/O concluída,
Interrupção 21: Dispositivo=TECLADO, PID=11, Tempo=60, Mensagem=Operação de I/O concluída,
```

Relatório Completo de Métricas

```
        RELATÓRIO COMPLETO DE MÉTRICAS

===== TEMPO DE RETORNO POR PROCESSO =====
P0: 30 ticks
P1: 30 ticks
P2: 30 ticks
P3: 30 ticks
P4: 30 ticks
P5: 30 ticks
P6: 30 ticks
P7: 30 ticks
P8: 30 ticks
P9: 30 ticks
P10: 30 ticks
P11: 30 ticks
P12: 30 ticks
P13: 30 ticks

Média: 30,00 ticks
========================================

===== TEMPO DE ESPERA EM PRONTO =====
P0: 0 ticks
P1: 0 ticks
```

7. Referências

- SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Operating System Concepts**. Hoboken: Wiley, 2018.
- TANENBAUM, Andrew S.; BOS, Herbert. **Modern Operating Systems**. 4. ed. Boston: Pearson, 2015.
- STALLINGS, William. **Operating Systems: Internals and Design Principles**. 9. ed. Upper Saddle River: Pearson, 2018.
- MICROSOFT. **Documentação oficial .NET e C#**. Disponível em: https://learn.microsoft.com/dotnet. Acesso em: 30 nov. 2025.
- MICROSOFT. **Documentação WPF (Windows Presentation Foundation)**. Disponível em: https://learn.microsoft.com/dotnet/desktop/wpf. Acesso em: 30 nov. 2025.
- STALLINGS, William. **Computer Organization and Architecture**. 11. ed. Boston: Pearson, 2020.
- TANENBAUM, Andrew S.; WOODHULL, Albert S. **Operating Systems: Design and Implementation**. 3. ed. Upper Saddle River: Prentice Hall, 2006.
- PATTERSON, David A.; HENNESSY, John L. **Computer Organization and Design: The Hardware/Software Interface**. 5. ed. Burlington: Morgan Kaufmann, 2014.
- NUTT, Gary J. **Operating Systems: A Modern Perspective**. 2. ed. Boston: Addison-Wesley, 2004.
- BOVET, Daniel P.; CESATI, Marco. **Understanding the Linux Kernel**. 3. ed. Sebastopol: O'Reilly Media, 2005.