

# Trabalho de Sistemas Operacionais — Simulador de Escalonamento e Gerência de Memória

---

## Objetivos de Aprendizagem

1. Implementar e comparar algoritmos de **escalonamento de processos** em um simulador de eventos discretos.
  2. Explorar o impacto de **sobrecargas, deadlines e prioridades** no desempenho do sistema.
  3. (Bônus) Investigar a influência da **memória virtual** no escalonamento e no desempenho global.
  4. (Avançado) Entender o princípio de justiça do **escalonador CFS (Completely Fair Scheduler)** do Linux.
- 

## Escopo do Trabalho

O aluno (ou grupo de até 4) deverá desenvolver um **simulador de escalonamento de processos** que suporte os algoritmos abaixo, visualize a execução (Gantt) e gere métricas quantitativas.

### Algoritmos obrigatórios

1. **FIFO / FCFS** – First Come, First Served (não preemptivo)
  2. **SJF** – Shortest Job First (não preemptivo)
  3. **Round-Robin** – quantum fixo (preemptivo)
  4. **EDF** – Earliest Deadline First (preemptivo)
  5. **CFS-Sim** – versão simplificada do *Completely Fair Scheduler* (preemptivo, justo)
- 

## Entradas do simulador

- Conjunto de processos **P[i]** com atributos:  
Parâmetros globais:

Atributo	Descrição
id	Identificador único
chegada	instante de chegada
execução	tempo total de CPU
prioridade	inteiro (1 = maior prioridade)

`deadline` instante máximo desejado (absoluto)

`Num_paginas` (bônus) páginas de memória virtual

`quantum` (u.t.) — para os algoritmos preemptivos

`sobrecarga_contexto` (u.t.) — tempo gasto em cada troca de contexto

`custo_disco` (u.t.) — atraso em *page fault* (bônus)

`seed` — garante determinismo para sorteios de página

---

## Regras Gerais de Execução

1. CPU **single-core**.
  2. Fila de prontos recebe processos à medida que chegam.
  3. Troca de contexto adiciona `sobrecarga_contexto` como CPU ociosa.
  4. Caso nenhuma tarefa esteja pronta, o tempo avança até a próxima chegada.
  5. Se `termino > deadline`, o processo é marcado como **estourado**.
  6. Métricas individuais:
    - `turnaround = termino - chegada`
    - `espera = turnaround - execucao`
  7. Métricas globais: médias, throughput, % de CPU ociosa, número de preempções.
- 

## Saídas Obrigatórias

- **Gráfico de Gantt:**
    - verde → execução
    - vermelho → sobrecarga de contexto
    - cinza → estouro de deadline
    - linha vertical → deadline absoluto
  - **Tabela final:** `chegada`, `execucao`, `deadline`, `prioridade`, `inicio(s)`, `termino`, `espera`, `turnaround`, `deadline_ok?`.
  - **Resumo quantitativo:** médias, throughput, % ociosidade, total de trocas de contexto.
  - **Modo passo-a-passo** (opcional, p/ demonstração): atraso visual entre eventos.
- 

## Algoritmos a Implementar

### FIFO / FCFS

- Ordem de chegada.
- Não preemptivo.

## SJF

- Escolhe menor tempo de execução restante.
- Não preemptivo.

## Round-Robin

- Quantum fixo (**quantum**).
- Cada preempção conta troca de contexto.
- Fila circular.

## EDF (Earliest Deadline First)

- Escolhe processo com **menor deadline absoluto**.
- Preemptivo: se chega um com deadline menor, interrompe o atual.
- Mantém fairness mínima pela prioridade em empates.

## CFS-Sim — *Completely Fair Scheduler Simplificado*

Versão inspirada no escalonador real do Linux (CFS), com foco em justiça e tempo virtual.

### Conceito

Cada processo mantém um **tempo virtual** (**vruntime**) que acumula de acordo com o tempo já executado e sua prioridade (*nice*).

$$vruntime_i = vruntime_i + \Delta t \times w(prioridade_i)$$

onde

$$w(prioridade) = 1.25^{(prioridade-1)}$$

### Regras

1. Quando um processo chega, **vruntime** = **tempo\_atual**.
2. A cada fatia de CPU ( $\Delta t$ ), o **vruntime** do processo ativo aumenta segundo sua prioridade.
3. O próximo processo a executar é o de **menor vruntime**.
4. Se novos processos chegam, entram na estrutura ordenada por **vruntime**.
5. Preempção ocorre quando outro processo tem **vruntime** menor que o atual.
6. **Sem quantum fixo**: o tempo de cada fatia emerge do reequilíbrio entre **vruntimes**.
7. Ao final, métricas iguais às demais políticas.

## Estrutura sugerida

```
class Processo:
    id, chegada, execucao, restante, prioridade
    vruntime = 0
```

## Pseudocódigo resumido

```
loop:
    adicionar novos processos com vruntime = tempo_atual
    if CPU livre:
        P = min(prontos, key=vruntime)
        executar P por  $\Delta t$  (mínimo entre resto e fatia_granular)
        P.vruntime +=  $\Delta t * w(P.prioridade)$ 
        if P terminou: remover; else reinserir
```

## Observações

- O CFS-Sim **não usa filas de prioridade fixas**, mas **balanceia justiça temporal**.
  - Representa a ideia de “CPU proporcional” usada no kernel Linux.
  - No Gantt, cada processo aparece intercalando execuções curtas e frequentes.
  - Faz parte do trabalho estudar/interpretar este código.
- 

# (Bônus) Módulo de Memória Virtual

## Modelo simplificado

- RAM: 200 KB (50 frames de 4 KB).
  - Cada processo: até 10 páginas.
  - Políticas: FIFO ou LRU.
  - *Page fault* → bloqueio por `custo_disco` u.t.
  - O simulador deve:
    - Indicar quais páginas estão na RAM e quais no disco.
    - Gerar contagem total de *page faults* por processo.
    - Exibir bloqueios no Gantt (cor azul clara).
- 

## Formato de Entrada (JSON) (Opcional ser nesse formato)

```
{
  "quantum": 2,
  "sobrecarga_contexto": 1,
  "custo_disco": 3,
  "seed": 42,
  "processos": [
```

```
{ "id": "P1", "chegada": 0, "execucao": 5, "deadline": 8, "prioridade": 2, "num_paginas": 6 },  
  
{ "id": "P2", "chegada": 1, "execucao": 4, "deadline": 12, "prioridade": 1, "num_paginas": 4 },  
  
{ "id": "P3", "chegada": 3, "execucao": 6, "deadline": 14, "prioridade": 3, "num_paginas": 3 }  
]  
}
```

---

## Linha de Comando (sugerida)

- `simso --alg CFS --input casos/caso1.json --gantt out/caso1.png \`
  - `--log out/caso1.log --seed 42 --delay 150`
- 

## Testes Públicos (Opcional ser nesse formato)

### Caso Base (sem memória virtual)

```
quantum = 2; sobrecarga = 1  
P1: chegada=0, execucao=5, deadline=8, prioridade=2  
P2: chegada=1, execucao=4, deadline=12, prioridade=1  
P3: chegada=3, execucao=2, deadline=20, prioridade=3
```

Publicar Gantt esperado e métricas médias para cada algoritmo (inclusive CFS-Sim).

### Caso com Ociosidade

Processos chegam espaçados → valida tratamento de CPU idle.

### Caso com Memória Virtual (bônus)

Definir `num_paginas` e `custo_disco=3`; demonstrar *page faults* afetando escalonamento.

---

## Avaliação

Critério	Peso
Correção e fidelidade dos algoritmos	40 %
Visualização (Gantt, legendas, clareza)	20 %

Estrutura e documentação do simulador  
Análise comparativa quantitativa  
(Bônus) Memória virtual

20 %  
20 %  
+2,0 pts

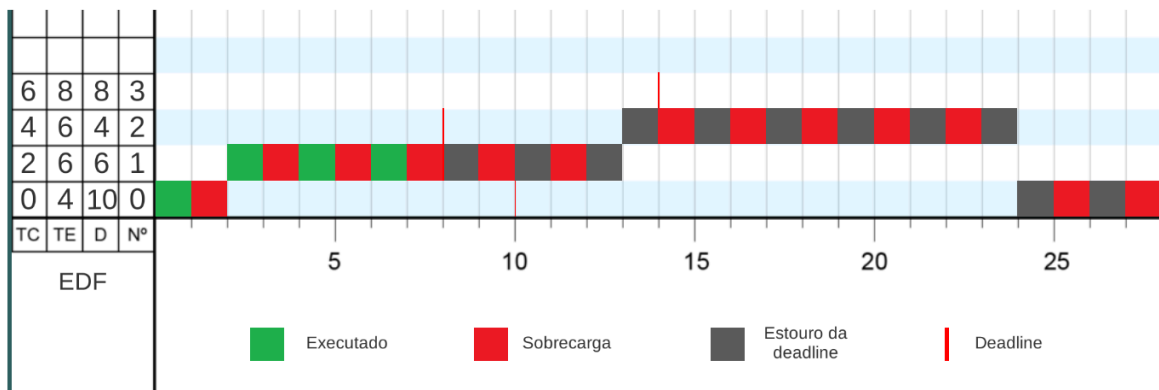
## Entregáveis

- Relatório (4-6 páginas) contendo:
  - design e decisões de implementação;
  - prints de Gantt's e tabelas;
  - análise quantitativa dos algoritmos;
  - (bônus) influência da VM nas métricas.
- Vídeo de até **6 min** mostrando:
  - execução dos algoritmos;
  - sobrecargas e deadlines visíveis;
  - (opcional) ativação do módulo de memória virtual.
- Apresentação com arguição em data a ser agendada.

## Visualização SUGERIDA

Processos:

Gráfico de Gantt



Memória RAM:

Gerência de Me...															
0	10	20	30	40	50	60	70	80	90						
1	11	21	31	41	51	61	71	81	91						
2	12	22	32	42	52	62	72	82	92						
3	13	23	33	43	53	63	73	83	93						
4	14	24	34	44	54	64	74	84	94						
5	15	25	35	45	55	65	75	85	95						
6	16	26	36	46	56	66	76	86	96						
7	17	27	37	47	57	67	77	87	97						
8	18	28	38	48	58	68	78	88	98						
9	19	29	39	49	59	69	79	89	99						

Disco:

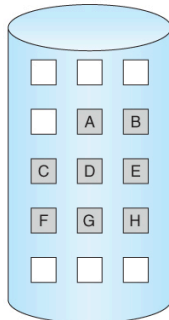


Tabela de Páginas

Invertida:

frame		valid-invalid bit	
0	4	v	
1		i	
2	6	v	
3		i	
4		i	
5	9	v	
6		i	
7		i	

page table