

Escalonamento de Processos em Sistemas de Tempo Real

Conceitos, Algoritmos e Análise de Escalonabilidade

Pedroso

29 de agosto de 2025

Sumário

- 1 Introdução ao Problema
- 2 Variáveis de um Processo
- 3 Algoritmo Rate Monotonic (RM)
- 4 Algoritmo Early Deadline First (EDF)
- 5 Implementação no Linux
- 6 Conclusão

Introdução ao Problema de Escalonamento

➤ O que é um Sistema de Tempo Real?

- ✓ Processos críticos precisam ser escalonados dentro de um limite de tempo (*deadline*).
- ✓ Exemplo: Freio ABS de um carro. Um atraso pode ser catastrófico.

➤ Tipos de Tarefas:

- ✓ **Tempo Real Rígido (Hard Real-Time):** Falhar ao escalonar o processo antes do *deadline* pode ser catastrófico.
- ✓ **Tempo Real Brando (Soft Real-Time):** Atrasos são indesejáveis, mas não causam falhas graves.

➤ O Problema do Escalonamento:

- ✓ Como alocar o tempo de CPU para que todas as tarefas cumpram seus prazos?

➤ Um sistema operacional de tempo compartilhado não pode ser usado para aplicações de tempo real!!!

Principais Sistemas Operacionais de Tempo Real

➤ FreeRTOS

- ✓ Um dos RTOS mais populares, de código aberto e amplamente usado em sistemas embarcados de pequeno porte.
- ✓ Oferece um kernel compacto e escalável, ideal para microcontroladores.

➤ VxWorks

- ✓ RTOS comercial robusto, com um histórico comprovado em aplicações críticas.
- ✓ Utilizado em setores como aeroespacial, defesa e robótica.

➤ QNX

- ✓ Um RTOS de microkernel, focado em segurança e alta disponibilidade.
- ✓ Frequentemente empregado na indústria automotiva (infotainment) e em sistemas industriais.

➤ RT-Linux (e outras variantes)

- ✓ Não é um RTOS puro, mas uma modificação do kernel Linux para adicionar capacidades de tempo real.
- ✓ Combina a flexibilidade e o vasto ecossistema do Linux com requisitos de tempo real.

Exemplo de Aplicações dos RTOS

➤ FreeRTOS

- ✓ **Aplicações:** Dispositivos de Internet das Coisas (IoT), sensores inteligentes, roteadores e sistemas embarcados de baixo consumo de energia.

➤ VxWorks

- ✓ **Aplicações:** É usado nos aviões *Boeing 787 Dreamliner* e *Airbus A350 XWB*. Na exploração espacial, foi utilizado pela NASA no *Mars Pathfinder* e no robô *Curiosity*.
- ✓ Além da NASA e da Boeing, é usado pela *Lockheed Martin* em sistemas de defesa.

➤ QNX

- ✓ **Aplicações:** Muito usado na indústria automotiva. Exemplos: *Porsche Panamera* e de muitos veículos da *Ford SYNC*.
- ✓ **Empresas usuárias:** *Audi*, *Ford*, *General Electric (GE)* em sistemas de controle industrial.

➤ Linux

- ✓ **Aplicações:** O Linux foi empregado nos computadores de voo redundantes a bordo dos foguetes **Falcon 9** e das cápsulas **Dragon**. Encontrado também em equipamentos de automação industrial da *Siemens* e em sistemas robóticos.

Variáveis de um Processo em Tempo Real

- ▶ **Tempo de Execução (C_i):** Tempo de CPU necessário para a tarefa i ser concluída.
 - ✓ Exemplo: $C_1 = 2\text{ms}$.
- ▶ **Período (T_i):** Intervalo entre duas chegadas consecutivas da tarefa i .
 - ✓ Exemplo: $T_1 = 10\text{ms}$.
- ▶ **Deadline (D_i):** Prazo final para a conclusão de uma instância de tarefa. Geralmente $D_i = T_i$.
 - ✓ Exemplo: $D_1 = 10\text{ms}$.
- ▶ **Utilização (U_i):** Fração do tempo de CPU que a tarefa demanda.
 - ✓ Fórmula: $U_i = C_i / T_i$.
 - ✓ Exemplo: $U_1 = 2\text{ms} / 10\text{ms} = 0,2$.
 - ✓ Neste exemplo, a tarefa 1 ocupa 20% da CPU

Utilização Total e Escalonabilidade

- **Utilização Total (U_{total}):** Soma das utilizações de todas as tarefas.
 - ✓ Fórmula: $U_{total} = \sum_{i=1}^n (C_i / T_i)$.
- **Condição Necessária para Escalonabilidade:**
 - ✓ A utilização total deve ser menor ou igual a 1.
 - ✓ $U_{total} < 1$
 - ✓ **Atenção:** Esta é uma condição necessária, mas **não suficiente!**

Algoritmo Rate Monotonic (RM)

- Algoritmo **estático** e **preemptivo**.
- **Critério de Prioridade:** Quanto menor o período (T_i), maior a prioridade.
- *Rate* se refere à taxa de repetição ($1/T_i$).
- O termo *Monotonic* refere-se ao fato de que as prioridades das tarefas são fixas e não mudam durante toda a execução do sistema.

Análise de Escalonabilidade para RM

➤ Teste de Utilização de Liu e Layland:

- ✓ Se $U_{total} \leq n(2^{1/n} - 1)$, o sistema é **garantidamente escalonável**.
- ✓ n é o número de tarefas.

➤ Valores do Limite Superior:

- ✓ $n = 1 \rightarrow 100\%$
- ✓ $n = 2 \rightarrow 82,8\%$
- ✓ $n = 3 \rightarrow 77,9\%$
- ✓ $n \rightarrow \infty \rightarrow \approx 69,3\%$

- Se U_{total} estiver entre o limite e 100%, o sistema **pode ou não** ser escalonável.
- É necessário realizar a análise de escalonabilidade utilizando o diagrama de Grantt.
- A *Análise de Tempo de Resposta* (RTA - Response Time Analysis) é um algoritmo que oferece precisão e rigor matemático da escalonabilidade, determinando tempo de resposta o pior cenário possível.

Análise de Escalonabilidade para RM

- Considere um sistema com duas tarefas:

Dados

Tarefa 1: $C_1 = 2\text{ms}$, $T_1 = 5\text{ms}$

Tarefa 2: $C_2 = 4\text{ms}$, $T_2 = 10\text{ms}$

- Anote a análise com diagrama de Grantt realizada durante a aula!!

Algoritmo Early Deadline First (EDF)

- Algoritmo **dinâmico** e **preemptivo**.
- O princípio do EDF é muito direto: em qualquer instante, o escalonador executa a tarefa com o deadline mais próximo.
- A prioridade de uma tarefa não é fixa. Ela muda dinamicamente à medida que os prazos se aproximam.
- Cada tarefa em um sistema tem um prazo para ser concluída. O EDF mantém uma fila de tarefas prontas para execução, ordenada pelo tempo restante até seu deadline.
- O SO sempre executa a tarefa que está no topo dessa fila, ou seja, aquela cujo deadline está mais próximo.
- Sempre que uma nova tarefa chega ou uma tarefa termina, a fila é reordenada para garantir que a próxima tarefa a ser executada seja a com o deadline mais próximo.

Exemplo de Escalonamento (EDF)

Tarefa 1: $C_1 = 2\text{ms}$, $T_1 = 5\text{ms}$, $D_1 = 5\text{ms}$

Tarefa 2: $C_2 = 4\text{ms}$, $T_2 = 10\text{ms}$, $D_2 = 10\text{ms}$

► Teste de Utilização:

- ✓ Um sistema é escalonável pelo EDF se, e somente se, a utilização total for menor ou igual a 1.
- ✓ $U_{total} \leq 1$
- ✓ **Essa condição é suficiente e necessária.**
- ✓ Essa regra vale somente para o caso onde o *deadline* é igual ao período da tarefa.
- ✓ Caso o *deadline* seja menor que o período, deve ser realizada a *Análise de Demanda de Tempo de CPU (Processor Demand Analysis)*.

► Vantagem do EDF:

- ✓ É um algoritmo **ótimo** para sistemas de tarefas periódicas.
- ✓ Permite utilizar a capacidade total da CPU (100%).

Exercício: Realize a análise de escalonabilidade

➤ Definição das Tarefas:

- ✓ **Tarefa 1:** $C_1 = 10$ ms, $T_1 = 50$ ms
- ✓ **Tarefa 2:** $C_2 = 30$ ms, $T_2 = 100$ ms
- ✓ **Tarefa 3:** $C_3 = 30$ ms, $T_3 = 200$ ms

➤ Cálculo da Utilização Total:

- ✓ $U_1 = 10/50 = 0,2$
- ✓ $U_2 = 30/100 = 0,3$
- ✓ $U_3 = 30/200 = 0,15$
- ✓ $U_{total} = 0,2 + 0,3 + 0,15 = 0,65$ (65%)

Análise de Escalonabilidade para os algoritmos RM e EDF

► Teste com RM (Liu e Layland):

- ✓ Limite ($n = 3$): $U_{limite} = 3(2^{1/3} - 1) \approx 0.779$ (77.9%)
- ✓ Condição: $U_{total} \leq U_{limite}$
- ✓ $0,7 \leq 0,779 \rightarrow$ **APROVADO**
- ✓ O sistema é escalonável pelo RM.

► Teste com EDF:

- ✓ Condição: $U_{total} \leq 1$
- ✓ $0,7 \leq 1 \rightarrow$ **APROVADO**
- ✓ O sistema é escalonável pelo EDF.

► Para ambos os casos faça a análise utilizando o Diagrama de Grantt.

Considerações sobre interrupções

- A ocorrência de uma interrupção, por sua natureza, interrompe a execução da tarefa que está em andamento para que a rotina de tratamento seja executada. Esse tempo de processamento consome tempo de CPU que, de outra forma, seria dedicado às tarefas. Se esse tempo não for considerado, a análise de escalabilidade original, pode se tornar inválida. Alternativas:
 - ✓ Considerar as rotinas de tratamento de interrupção como uma tarefa de maior prioridade: isso garante a análise.
 - ✓ Incorporar o tempo de interrupção no uso da CPU das tarefas. Solução simples, mas com muitos problemas de implementação prática, ex. podem ocorrer múltiplas interrupções ao longo da execução de uma tarefa.

Escalonamento de Tempo Real no Linux

- O kernel Linux usa **políticas de escalonamento** para gerenciar o tempo do processador.
- Implementando o **Rate Monotonic (RM)**:
 - ✓ RM é um algoritmo de prioridade fixa.
 - ✓ Use a política **SCHED_FIFO** ou **SCHED_RR**.
 - ✓ **Como fazer:** Atribua prioridades fixas (de 1 a 99) para cada tarefa, onde uma prioridade mais alta corresponde a um período mais curto, seguindo a regra do RM.
- Implementando o **Early Deadline First (EDF)**:
 - ✓ EDF é um algoritmo de prioridade dinâmica.
 - ✓ Use a política **SCHED_DEADLINE**.
 - ✓ **Como fazer:** Defina os parâmetros de tempo real da tarefa:
 - ✓ **runtime:** tempo de CPU necessário
 - ✓ **deadline:** prazo absoluto para a execução
 - ✓ **period:** intervalo de repetição da tarefa
 - ✓ O kernel gerencia as prioridades dinamicamente com base nesses parâmetros.

► Resumo dos Pontos Principais:

- ✓ O sistema de tempo real busca cumprir os deadlines das tarefas.
- ✓ Diferença entre escalonamento estático (RM) e dinâmico (EDF).
- ✓ RM é simples, mas com limite de utilização relativamente mais baixo.
- ✓ O RM é subótimo ($U_{limite} < 1$) e o EDF é ótimo ($U_{limite} = 1$)
- ✓ Com o RM o Engenheiro necessita realizar análise de escalonabilidade, de preferência, utilizando Análise de Tempo de Resposta (Response Time Analysis).
- ✓ Com EDF se o *deadline* for igual ao período, então a análise é mais simples ($U_{total} < 1$).
- ✓ Com EDF se o *deadline* for diferente do período, então o Engenheiro deve fazer a Análise de Demanda de Tempo (Processor Demand Analysis)
- ✓ EDF é ótimo, mas com maior complexidade computacional.