Entrada/Saída em Sistemas Operacionais

Sistemas Operacionais Embarcados / TE355

João Gabriel Pazinato de Bittencourt

18 de novembro de 2025

Dispositivos de E/S

Gestão de E/S

- **O que são?**
  - Dispositivos de entrada: mouse, teclado, microfone, câmera, joystick, etc.
  - Dispositivos de saída: monitor, impressora, fones de ouvido, etc.
- E/S é o elo entre o computador (CPU e memória) e o mundo externo. Um sistema de computação sem E/S é inútil.

O Desafio da Gestão de E/S

- O SO deve gerenciar uma grande diversidade de dispositivos, cada um com suas regras:
  - **Velocidade:** Discos rígidos (30–150 MB/s) vs. SSDs (500-3.500 MB/s).
  - **Modo de Transferência de Dados:** Dispositivos de **Bloco** (disco rígido, SSD) vs. Dispositivos de **Caractere** (teclado, mouse, impressora).
  - **Interface de Acesso:** Portas de E/S (data-in, data-out, status, control, etc.).
  - **Interrupções:** Rotinas de Tratamento de Interrupção (ISRs)

Interação Direta entre CPU e Periférico

- **Comunicação Low-Level:** O CPU se comunica com o periférico através de **Registradores** (portas E/S).
- **Exemplo: Processo de Leitura de um Bloco no Disco Rígido:**
  1. **1. Verificar Estado:** O CPU lê o **Registrador de Status** do disco para verificar que ele está livre.
  2. **2. Enviar Comando:** O CPU escreve o endereço do bloco (p. ex. trilha, cabeça e setor) e o comando "LER" no **Registrador de Comando**.
  3. **3. Aguardar:** O CPU aguarda uma **Interrupção** gerada pelo controlador do disco ou faz polling até o **Registrador de Status** mudar para "Pronto".
  4. **4. Receber Dados:** O CPU lê o dado do **Registrador de Dados**.

Interação Direta entre CPU e Periférico

- Cada dispositivo requer um conjunto de instruções diferente. Se o programador de aplicação tivesse que fazer isso, o desenvolvimento seria inviável.

Interação Direta entre CPU e Periférico

- Cada dispositivo requer um conjunto de instruções diferente. Se o programador de aplicação tivesse que fazer isso, o desenvolvimento seria inviável.
- **O Desafio Central:** Oferecer para o aplicativo/usuário uma interface de acesso ao hardware **abstrata e genérica**, representando as interações com os periféricos da forma mais simples e uniforme possível, e escondendo toda essa complexidade de hardware.

A Abstração de Dispositivos no Linux

Tudo é um Arquivo (*Everything is a File*)

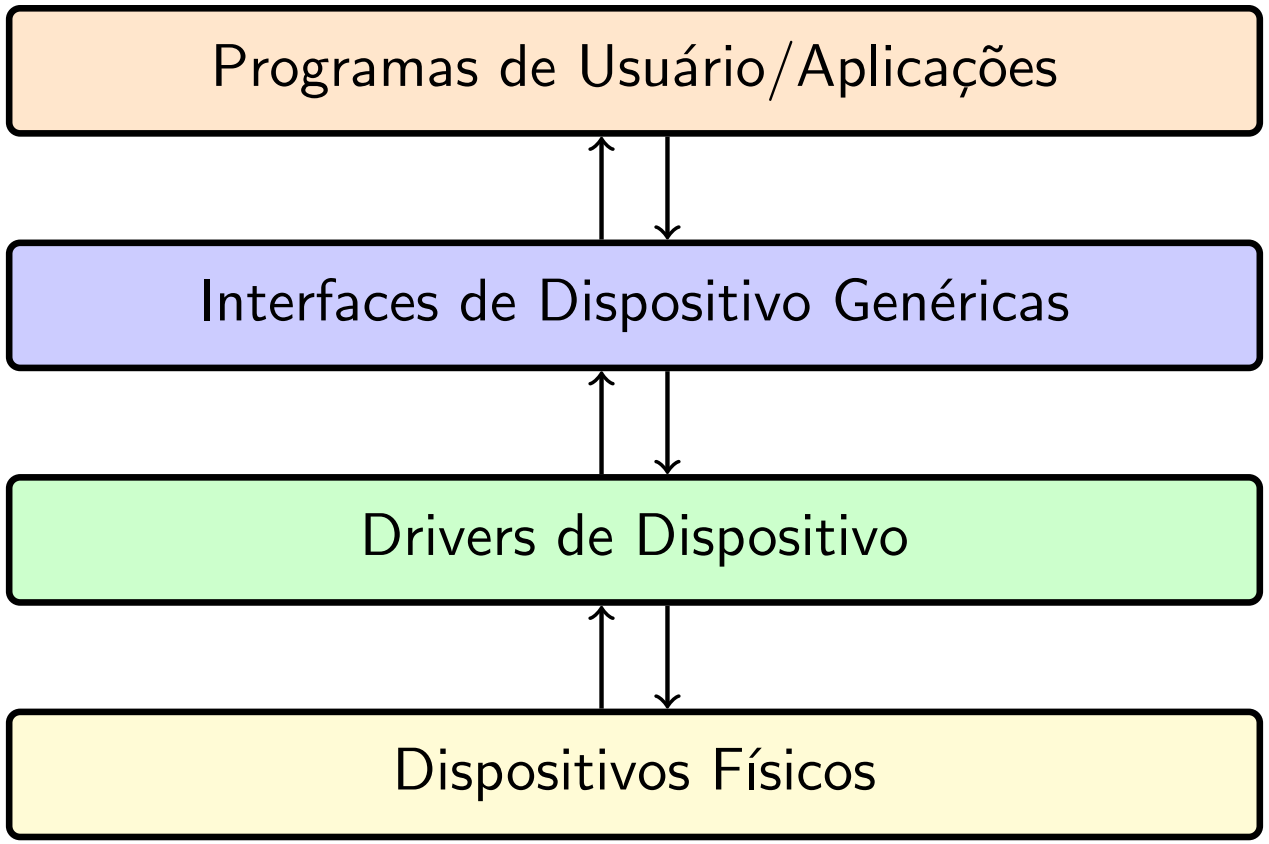
- É a forma que Sistemas Operacionais Unix simplificam a complexidade do hardware.
- Dispositivos de hardware (teclado, disco, portas seriais, etc.) são tratados como **arquivos**.
- Isso permite usar um conjunto **unificado** de funções de E/S para realizar quase todas as operações.

A Abstração de Dispositivos no Linux

Tudo é um Arquivo (*Everything is a File*)

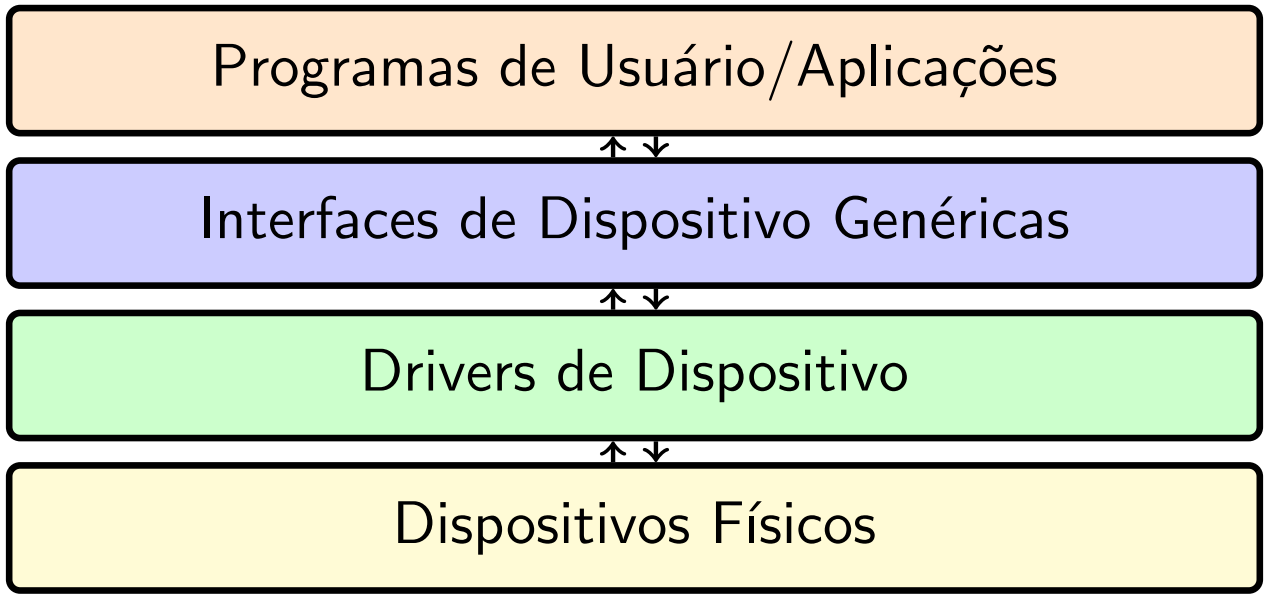
- É a forma que Sistemas Operacionais Unix simplificam a complexidade do hardware.
- Dispositivos de hardware (teclado, disco, portas seriais, etc.) são tratados como **arquivos**.
- Isso permite usar um conjunto **unificado** de funções de E/S para realizar quase todas as operações.
- **Exemplo prático no Raspberry Pi (Terminal Linux)**

Camadas de Abstração do Software de E/S



Propósito das Camadas de Abstração

- Cada camada da arquitetura possui um propósito claro e interage com as camadas adjacentes sob formas bem definidas
- **Modularidade:** A mudança de uma parte (ex: trocar ou instalar um driver) não afeta as demais (ex: a aplicação do usuário).
- **Reuso de Código:** Camadas superiores apresentam funções que podem ser usadas para manipular vários dispositivos diferentes.



Drivers de Dispositivo

- **Driver:** Código específico do núcleo que "conhece" os **registradores** e o **protocolo de comunicação** (e.g., UART, I2C, SPI) de um hardware específico.
- **Principais Funções dos Drivers:**
  - Controle e Configuração.
  - Tradução/Transferência de dados.
  - Tratamento de Interrupções.

Drivers de Dispositivo

- **Driver:** Código específico do núcleo que "conhece" os **registradores** e o **protocolo de comunicação** (e.g., UART, I2C, SPI) de um hardware específico.
- **Principais Funções dos Drivers:**
  - Controle e Configuração.
  - Tradução/Transferência de dados.
  - Tratamento de Interrupções.
- **Interrupções:** É o método preferível de comunicação entre hardware e SO.
  - O hardware envia um sinal (IRQ) para o CPU quando uma tarefa é concluída.
  - Isso permite que o CPU trabalhe em outras tarefas, ao invés de ficar em polling constante.

Interfaces Genéricas

- Fornecem um conjunto padronizado de funções (abrir, fechar, ler, escrever) para **classes** de dispositivos.
- Exemplo: Todos os discos rígidos (Dispositivos de Bloco) usam o mesmo conjunto de comandos lógicos, independentemente da marca.

Interfaces Genéricas

- Fornecem um conjunto padronizado de funções (abrir, fechar, ler, escrever) para **classes** de dispositivos.
- Exemplo: Todos os discos rígidos (Dispositivos de Bloco) usam o mesmo conjunto de comandos lógicos, independentemente da marca.
- **Classes de Dispositivos:**
  - **Bloco:** Dados transferidos em blocos de tamanho fixo (discos, SSDs).
  - **Caractere:** Dados transferidos byte a byte (teclado, mouse, porta serial).
  - **Outras Classes Possíveis:** Dispositivos de Rede, de Vídeo, de Áudio, etc.

- **Chamadas de Sistema (System Calls):**
  - A camada mais alta do núcleo, acessível pela aplicação.
  - É a única forma que a aplicação tem para acessar o hardware, pois executa código em modo núcleo (privilegiado).
- `int open(const char *pathname, int flags);`
  - Abre o arquivo/dispositivo e retorna um **Descritor de Arquivo** (File Descriptor, FD).
- `ssize_t read(int fd, void *buf, size_t count);`
  - Tenta ler `count` bytes do FD e os armazena em `buf`. Retorna o número de bytes lidos.
- `ssize_t write(int fd, const void *buf, size_t count);`
  - Tenta escrever `count` bytes de `buf` no FD. Retorna o número de bytes escritos.
- `int close(int fd);`
  - Libera o FD e os recursos do sistema.

Exemplo de Fluxo de Controle

Tirar uma Foto Através do Aplicativo de Câmera

- **Downstream (Pedido):**
  1. **Aplicação:** O aplicativo de câmera realiza uma chamada de sistema (p. ex. `write()`) para solicitar a captura de imagem, transferindo o controle ao núcleo.
  2. **Interface genérica:** O núcleo encaminha a solicitação à interface de dispositivos de vídeo, que invoca uma sequência de funções implementadas pelo driver da câmera.
  3. **Driver:** O driver traduz a solicitação em comandos específicos, escrevendo nos registradores do controlador da câmera.
  4. **Hardware:** O controlador aciona a câmera e salva os dados da imagem em um buffer.

Exemplo de Fluxo de Controle

Tirar uma Foto Através do Aplicativo de Câmera

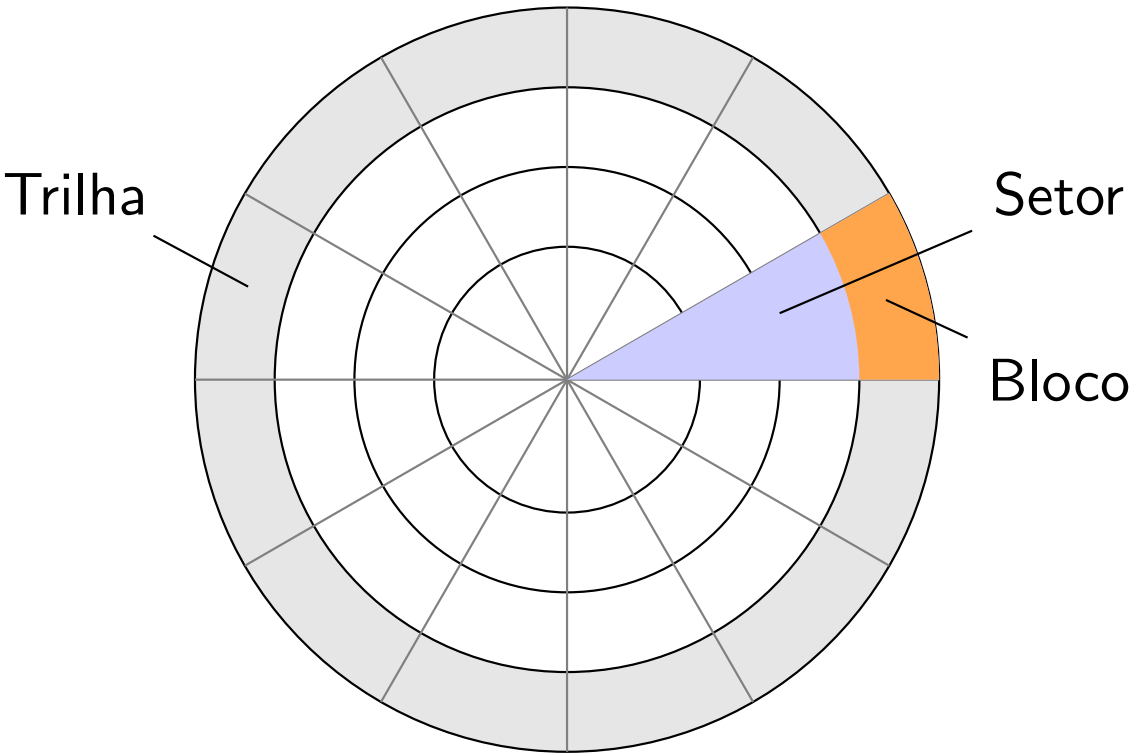
- **Upstream (Resposta):**
  1. **Hardware:** Ao finalizar a captura, o controlador gera um sinal de interrupção para o CPU, que inicia a Rotina de Tratamento de Interrupção (ISR) definida pelo driver.
  2. **Driver:** A execução da ISR transfere os dados do buffer à memória do sistema, e notifica o núcleo.
  3. **Aplicação:** O aplicativo é informado da conclusão da solicitação, e pode acessar a imagem em memória para armazená-la ou exibi-la no monitor.

Prática: Leitura e Escrita AD/DA no RPi

- **Hardware:** Raspberry Pi, conversor AD (externo) e módulo PWM (interno)
- **Leitura AD:** Exibir no terminal o sinal de tensão saindo de um potenciômetro (sensor).
- **Escrita DA:** Controlar o brilho de um LED (atuador) a partir da escrita de valores no pino PWM.

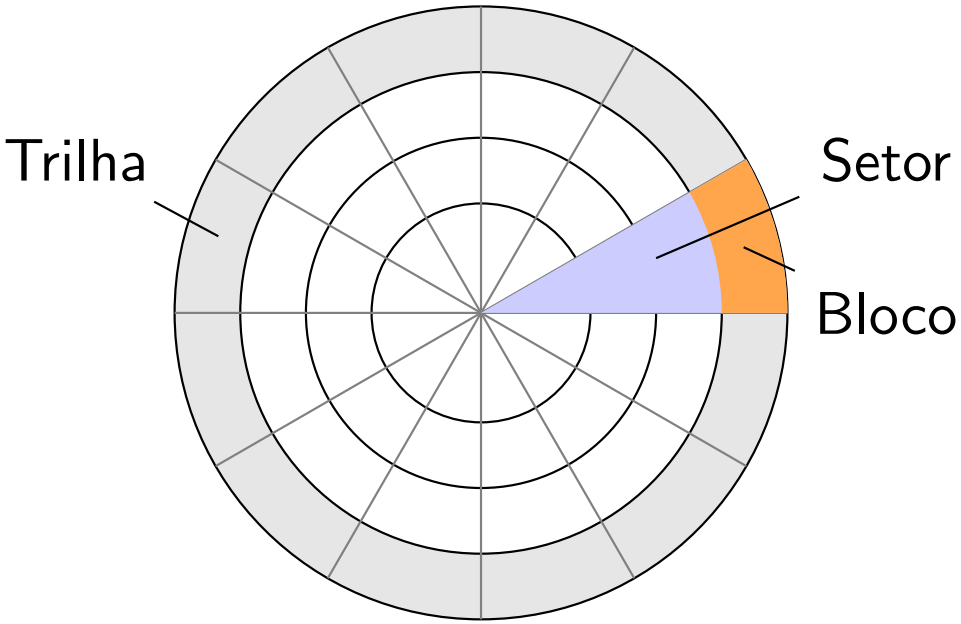
Discos Rígidos (HDD)

- **Importância:** Apesar do avanço dos SSDs, os HDDs continuam sendo uma forma crucial de armazenamento persistente, oferecendo o melhor custo-benefício por Terabyte.
- O HDD armazena dados em **blocos** de tamanho fixo (p. ex., 512 ou 4.096 bytes), organizados em **trilhas** e **setores**.



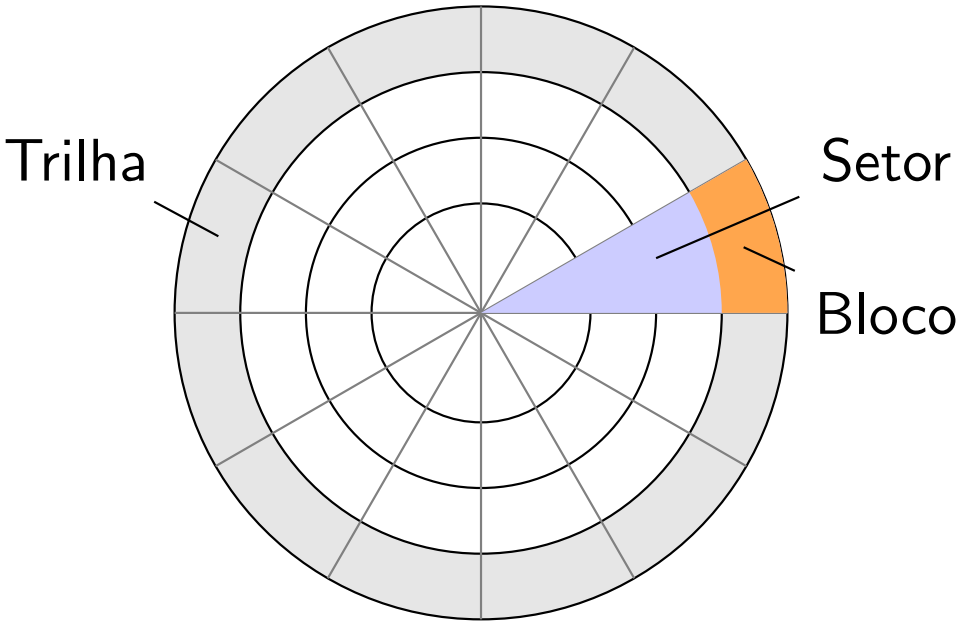
Discos Rígidos (HDD)

- **Componentes Físicos:**
  - **Discos metálicos:** São dispostos em pilha e giram em alta velocidade, divididos em trilhas e setores.
  - **Cabeças:** Usadas para leitura/escrita.
  - **Braço Atuador:** Move as cabeças sobre as trilhas.



Discos Rígidos (HDD)

- **Componentes Físicos:**
  - **Discos metálicos:** São dispostos em pilha e giram em alta velocidade, divididos em trilhas e setores.
  - **Cabeças:** Usadas para leitura/escrita.
  - **Braço Atuador:** Move as cabeças sobre as trilhas.



- **Problema Principal:** O acesso é **mecânico** e **lento**.

Tempo de Acesso

Necessidade de Escalonamento

- O tempo total para ler dados de um bloco é a soma de três componentes:
- **1. Tempo de Busca:** Tempo para mover o braço e a cabeça até a trilha correta.
  - **É o componente mais demorado** (em média 10 ms).
- **2. Latência Rotacional:** Tempo para o setor desejado girar até a cabeça (em média 5 ms).
- **3. Tempo de Transferência:** Tempo para transferir o bloco de dados (na ordem de microssegundos).

Tempo de Acesso

Necessidade de Escalonamento

- O tempo total para ler dados de um bloco é a soma de três componentes:
- **1. Tempo de Busca:** Tempo para mover o braço e a cabeça até a trilha correta.
  - **É o componente mais demorado** (em média 10 ms).
- **2. Latência Rotacional:** Tempo para o setor desejado girar até a cabeça (em média 5 ms).
- **3. Tempo de Transferência:** Tempo para transferir o bloco de dados (na ordem de microssegundos).
- **Conclusão:** O núcleo deve minimizar o **movimento total do braço** (Tempos de Busca) ao atender uma fila de requisições. Isso é feito pelo **Escalonador de Disco**.

Estratégias de Escalonamento de Disco

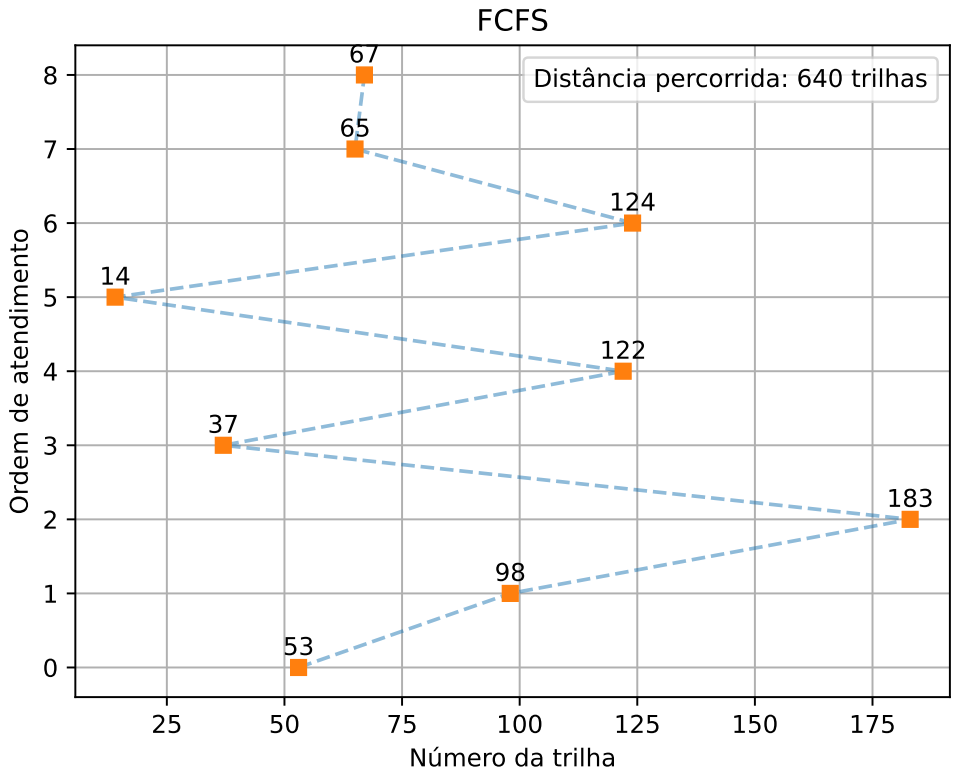
- **Objetivo:** Reduzir o **Tempo de Busca total** e, idealmente, garantir a **Justiça** (*fairness*) para todas as requisições.
- **Exemplo:** Suponha um disco hipotético de 200 trilhas. Inicialmente, a cabeça de leitura se encontra sobre a trilha 53 e a fila de requisições contém oito pedidos de acesso a blocos contidos nas seguintes trilhas:

98, 183, 37, 122, 14, 124, 65, 67

Estratégias de Escalonamento de Disco

Posição Inicial: 53, **Requisições:** 98, 183, 37, 122, 14, 124, 65, 67

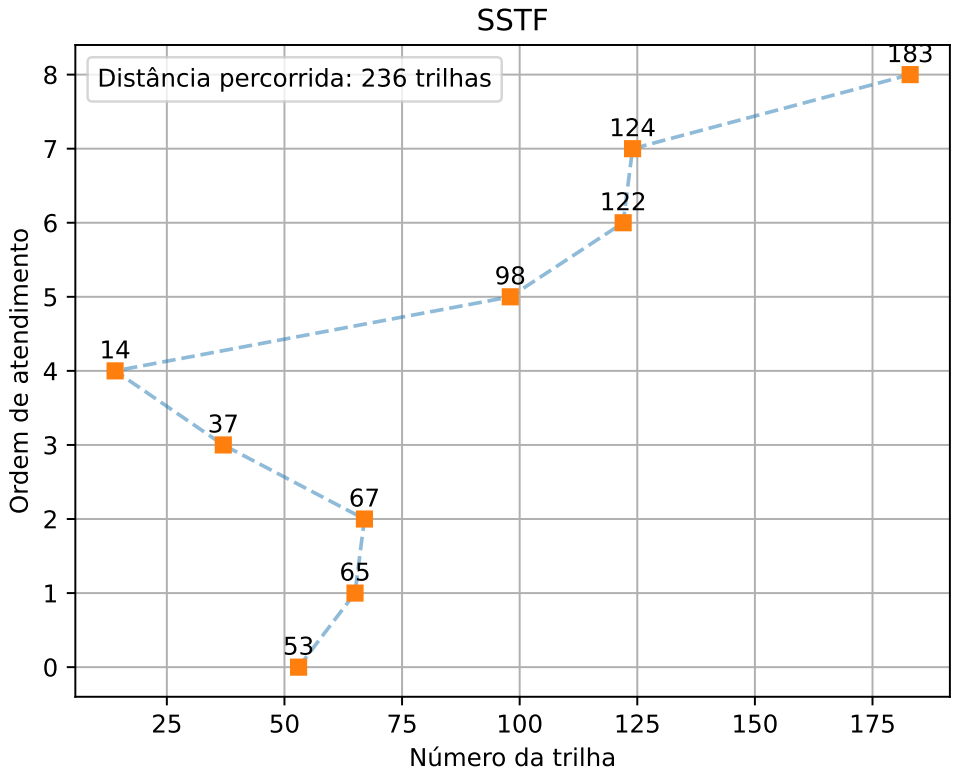
- **1. FCFS (First-Come, First-Served):**
  - Atende as requisições na ordem de chegada. É a solução mais simples, porém ineficiente.



Estratégias de Escalonamento de Disco

Posição Inicial: 53, **Requisições:** 98, 183, 37, 122, 14, 124, 65, 67

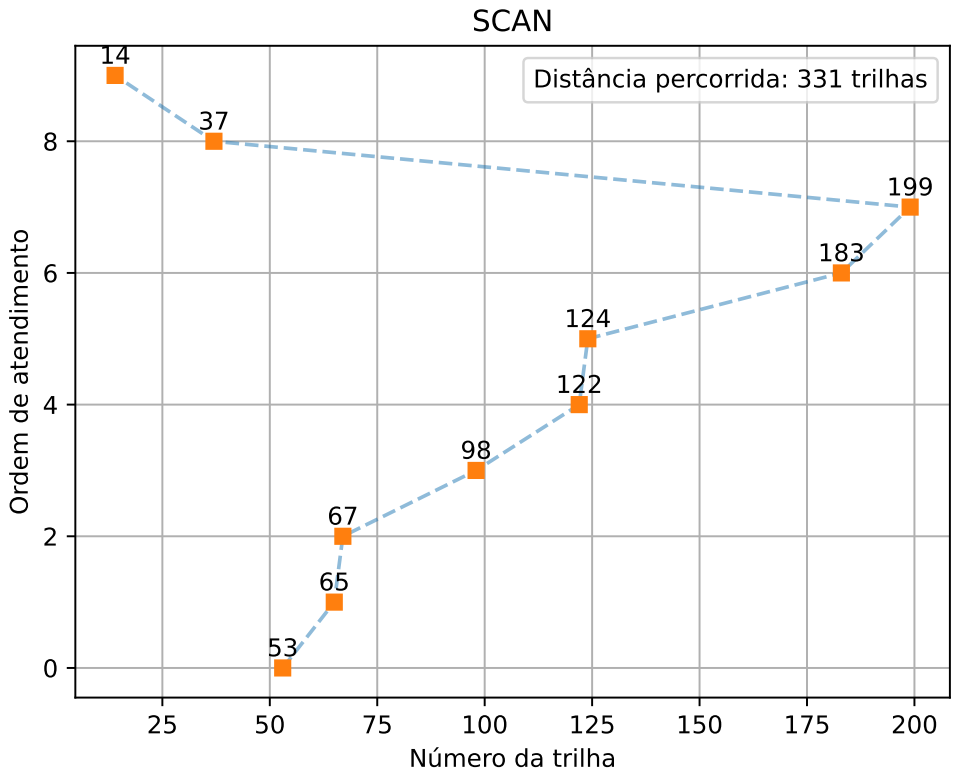
- **2. SSTF (Shortest Seek Time First):**
  - Atende a requisição **mais próxima** da posição atual da cabeça.
  - **Vantagem:** Reduz o tempo de busca.
  - **Desvantagem:** Risco de **inanição**.



Estratégias de Escalonamento de Disco

Posição Inicial: 53, **Requisições:** 98, 183, 37, 122, 14, 124, 65, 67

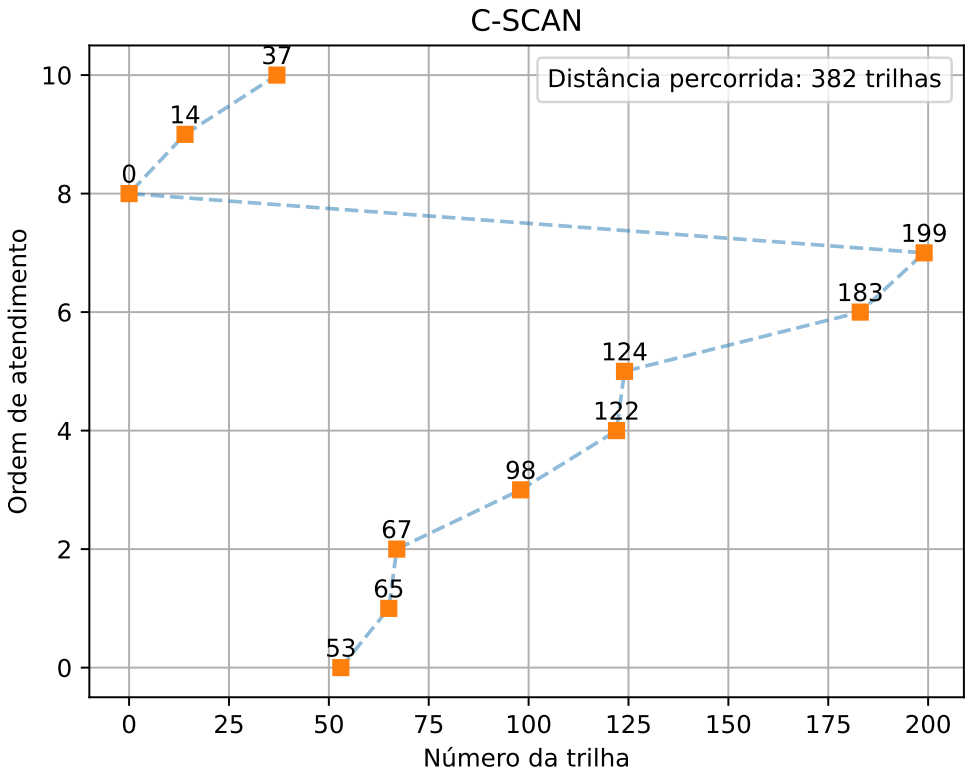
- **3. SCAN (Algoritmo do Elevador):**
  - Inicialmente, a cabeça se move em um **sentido único**, atendendo a todos os pedidos nessa direção.
  - Ao chegar a uma extremidade, ela **inverte** o sentido.
  - **Vantagem:** Reduz o risco de **inanição**.



Estratégias de Escalonamento de Disco

Posição Inicial: 53, **Requisições:** 98, 183, 37, 122, 14, 124, 65, 67

- **4. C-SCAN (Circular SCAN):**
  - A cabeça sempre se move em **uma único sentido** (p.ex., $0 \rightarrow 199$).
  - Ao chegar ao fim, ela retorna ao início sem atender pedidos na volta.
  - **Vantagem:** Oferece um tempo de resposta mais uniforme e evita ocorrências de **inanição**. É a base de muitos escalonadores modernos.



Exercício de Escalonamento

Considere um disco rígido com 200 trilhas numeradas de 0 a 199. A cabeça de leitura encontra-se inicialmente na trilha 135, e a fila de requisições pendentes é:

$$49, 64, 189, 103, 170, 21, 75$$

- Determine a ordem de atendimento das requisições segundo cada um dos seguintes algoritmos de escalonamento:
  1. SSTF (o pedido seguinte é sempre o mais próximo)
  2. SCAN (a cabeça faz varredura até um dos extremos e inverte o sentido)
  3. C-SCAN (a cabeça faz varredura até um dos extremos e retoma no outro extremo)
- Para cada algoritmo, calcule a distância total percorrida pela cabeça de leitura (em número de trilhas).

OBS: Considere que a cabeça se move inicialmente em direção às trilhas maiores para os algoritmos SCAN!

Referências

[1] Tanenbaum, A. S., Sistemas Operacionais Modernos.
[2] Maziero, C. A., Sistemas Operacionais: Conceitos e Mecanismos.