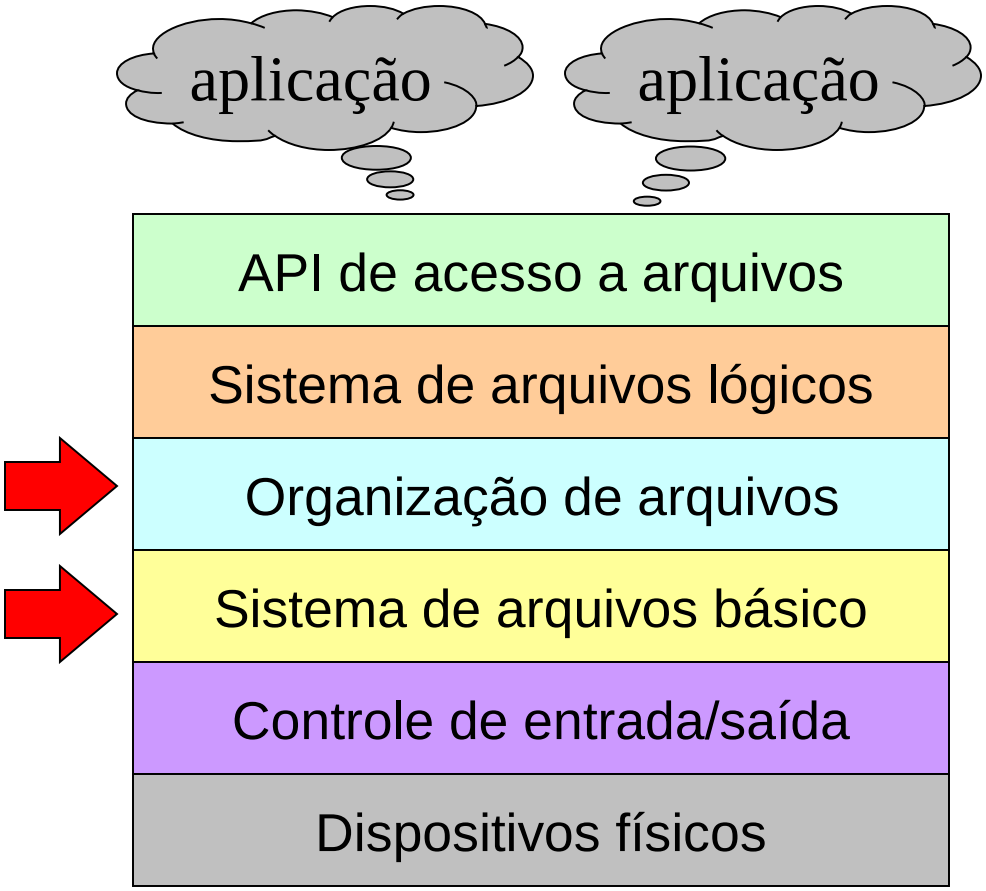
Sistema de arquivos

- Dispositivos com tecnologias variadas
  - CD-ROM, DAT, HD, Floppy, ZIP
  - SCSI, IDE, ATAPI, ...
  - sistemas de arquivos em rede
- Interfaces de acesso uniforme
  - visão homogênea dos dispositivos
  - transparência para as aplicações

Arquit. da gerência de arquivos



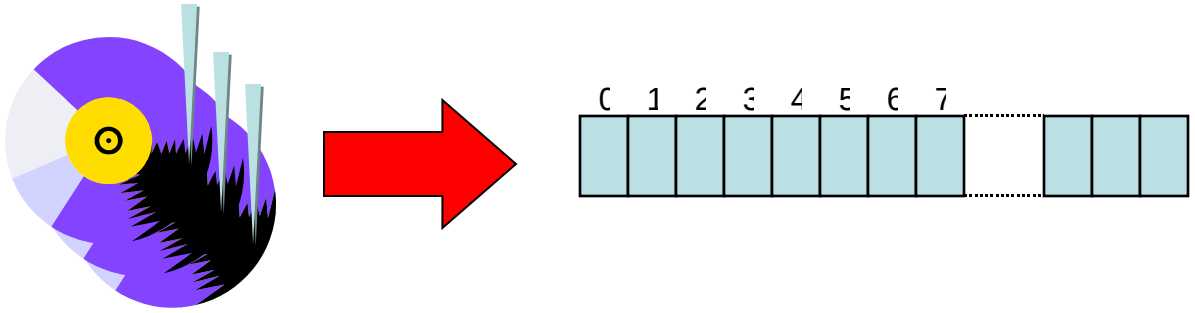
Dispositivos e drivers

- Dispositivo físico:
  - armazenamento dos dados
  - estruturados em blocos de bytes (~ 512 bytes)
  - CD-ROM, hard disk, floppy, fitas
- Driver de dispositivo:
  - acesso em baixo nível aos dispositivos
  - gerencia interrupções e DMA
  - mapeia acessos a trilhas/setores/cabeças em operações sobre portas de E/S do dispositivo

Visão dos dispositivos

- **Visão física**: cabeças, trilhas, setores
- **Visão lógica**: vetor de blocos idênticos

- Função do sistema de arquivos básico

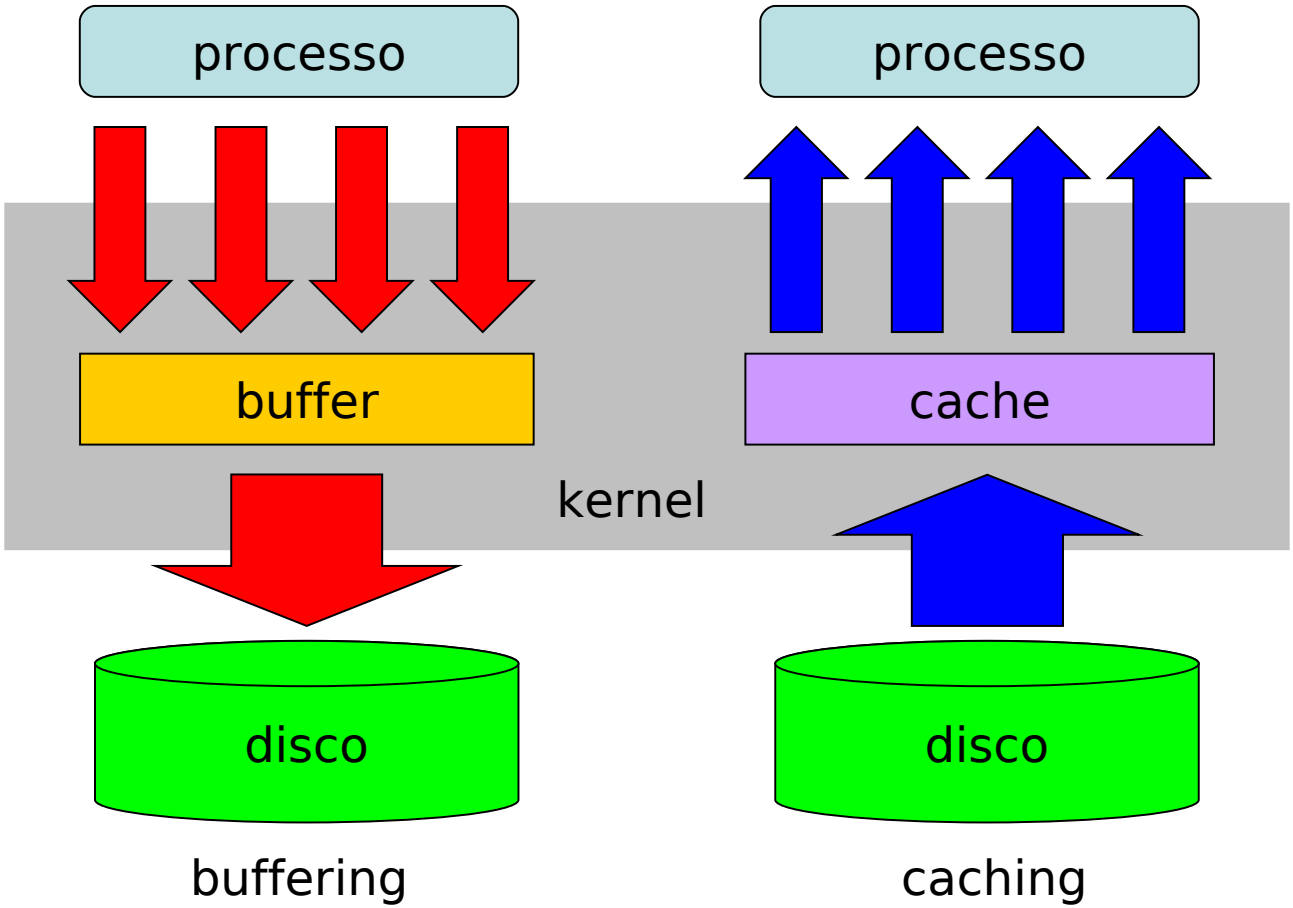


Visão física Visão lógica

Sistema de arquivos básico

- Aciona comandos de leitura/escrita nos drivers de dispositivos.
- Mostra o dispositivo como um vetor de blocos de mesmo tamanho.
  - **Blocos lógicos** entre 512 bytes e 8 Kbytes
- Pode efetuar *buffering* e *caching*:
  - *Buffering*: otimizar acessos reais em escrita.
  - *Caching*: otimizar acessos reais em leitura.

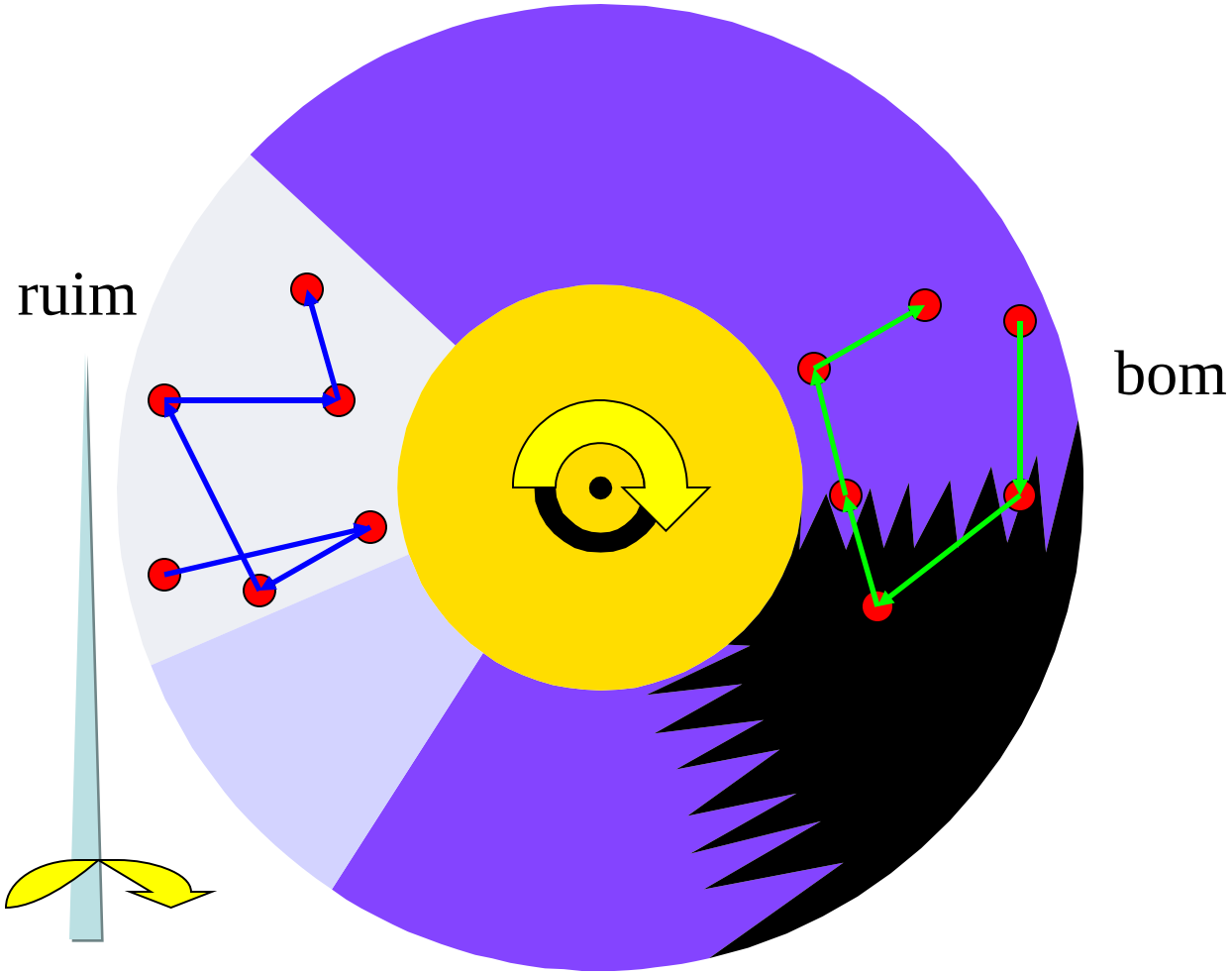
Buffering & caching



Escalonamento de disco

- Acesso ao disco por vários processos
  - processos acessam áreas distintas
  - o disco é um dispositivo LENTO
  - desempenho de I/O pode ser péssimo
- Acesso ao disco deve ser escalonado
  - escolher ordem de atendimento dos pedidos de acesso aos discos
  - buscar o melhor desempenho

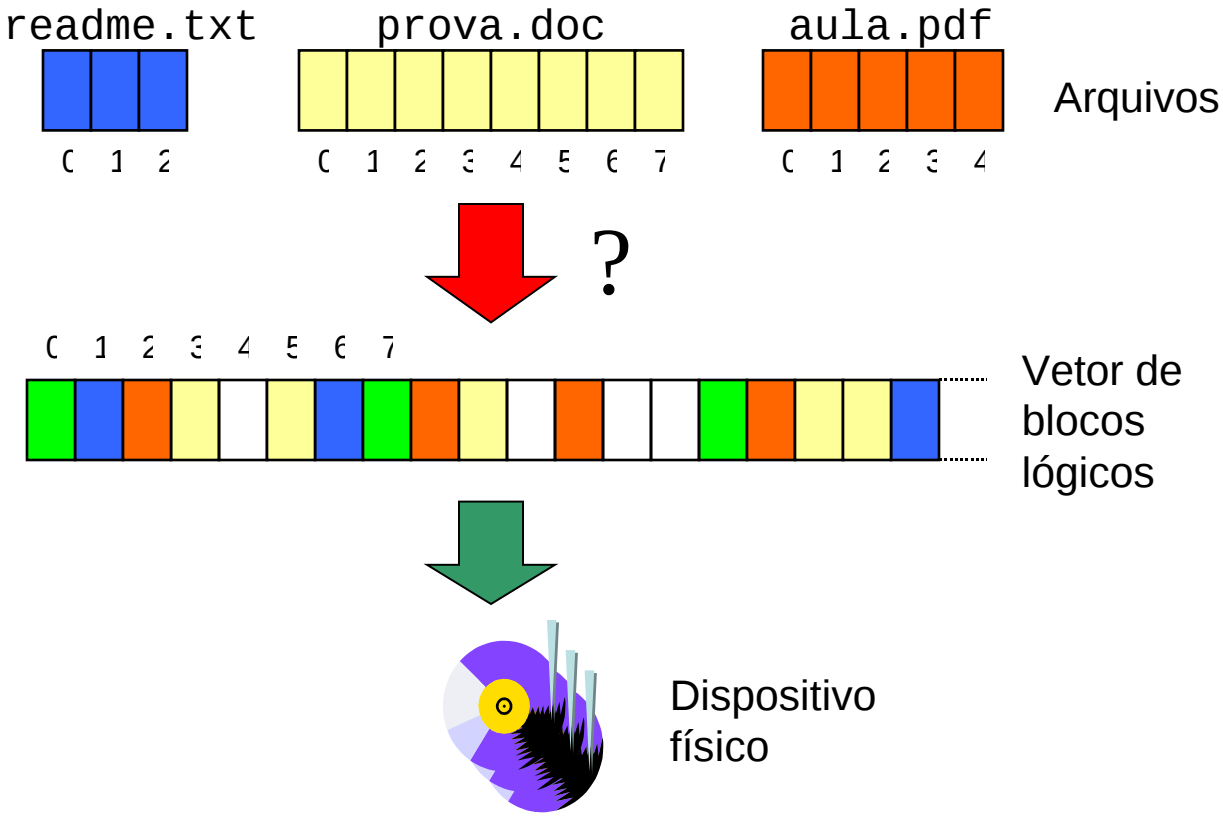
Exemplos de escalonamento



Organização de arquivos

- Problema:
  - Como armazenar diversos arquivos dentro de um único vetor de blocos lógicos ?
  - Cada arquivo também deve ser visto como uma seqüência de blocos lógicos.
- Restrições:
  - flexibilidade de alocação
  - rapidez de acesso (seqüencial e aleatório)
  - eficiência no uso do espaço real em disco

Organização de arquivos



Técnicas de alocação

- Formas de mapear os blocos dos arquivos em posições no vetor de blocos lógicos

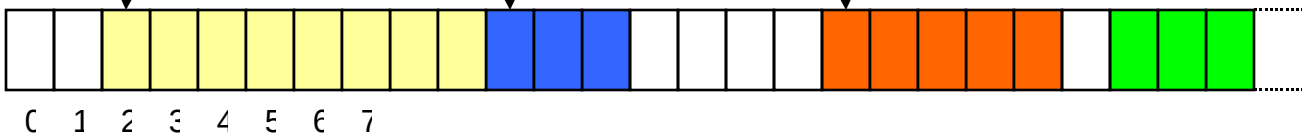
- Alocação **contígua** de arquivos
- Alocação em **listas encadeadas**
  - listas diretas ou **listas indexadas**
- Alocação **indexada**

Alocação contígua de arquivos

- Cada arquivo ocupa um conjunto de blocos lógicos consecutivos.
- Não há blocos vazios entre os blocos de um mesmo arquivo.
- Para cada arquivo, o diretório informa seu bloco de início e o nº de blocos.

Alocação contígua

arquivo	inicio	#blocos
readme.txt	010	003
prova.doc	002	008
Aula.pdf	017	005



Alocação contígua

- **Vantagens**:
  - **Simplicidade** de implementação.
  - **Rapidez** de acesso aos arquivos:
    - todos os blocos do arquivo estão próximos.
  - **Facilidade de acesso** seqüencial e aleatório:
    - **sequencial**: basta ler os blocos consecutivos
    - **aleatório**: posições internas podem ser facilmente calculadas a partir da posição do bloco inicial.

Alocação contígua

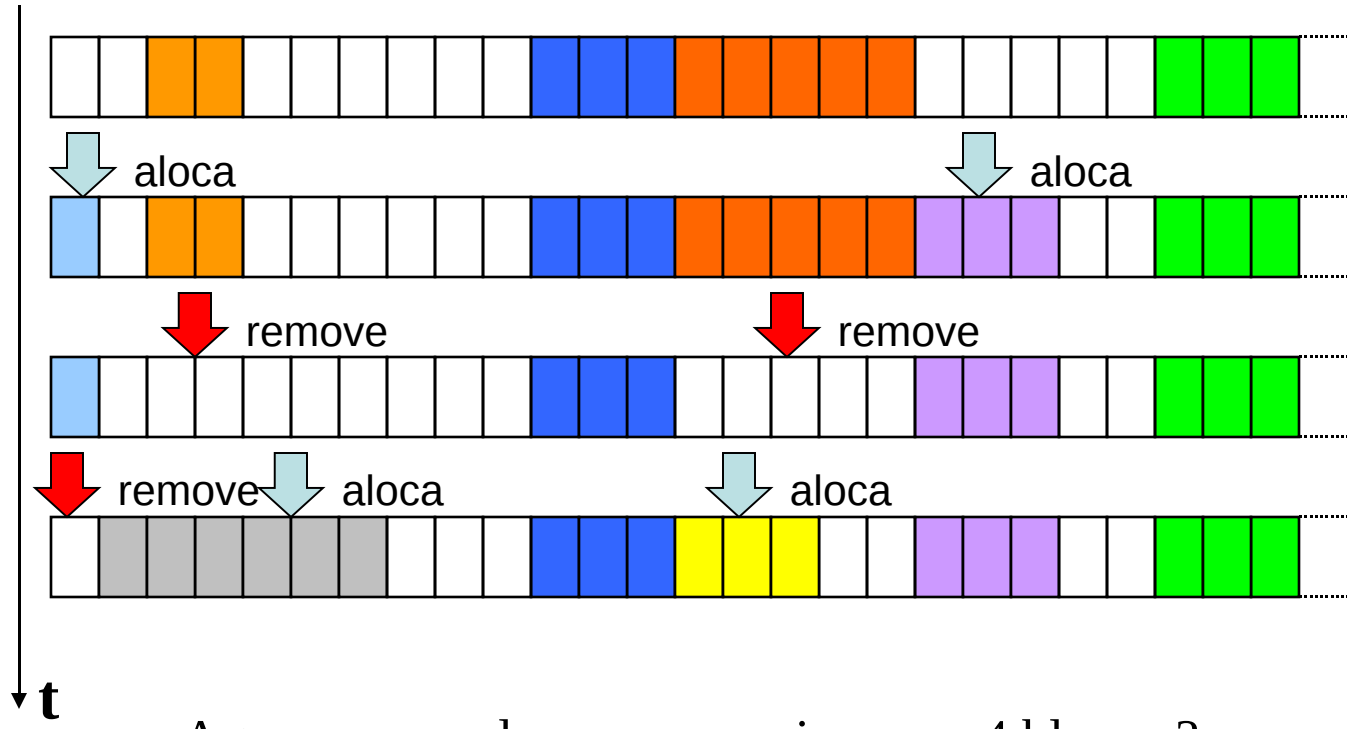
- Desvantagens:
  - Pouca flexibilidade no **crescimento** dos arquivos.
  - **Tamanho máximo** do arquivo deve ser conhecido no momento da alocação.
  - Ocorrência de **fragmentação externa**.
  - Necessidade de **desfragmentação** periódica.

Fragmentação externa

- Espaços vazios **entre** blocos de arquivos.
- À medida que o sistema evolui:
  - arquivos são criados e removidos
  - mais espaços vazios aparecem.
  - os espaços vazios ficam menores.

➡ Alocar novos arquivos torna-se difícil !

Evolução da fragmentação

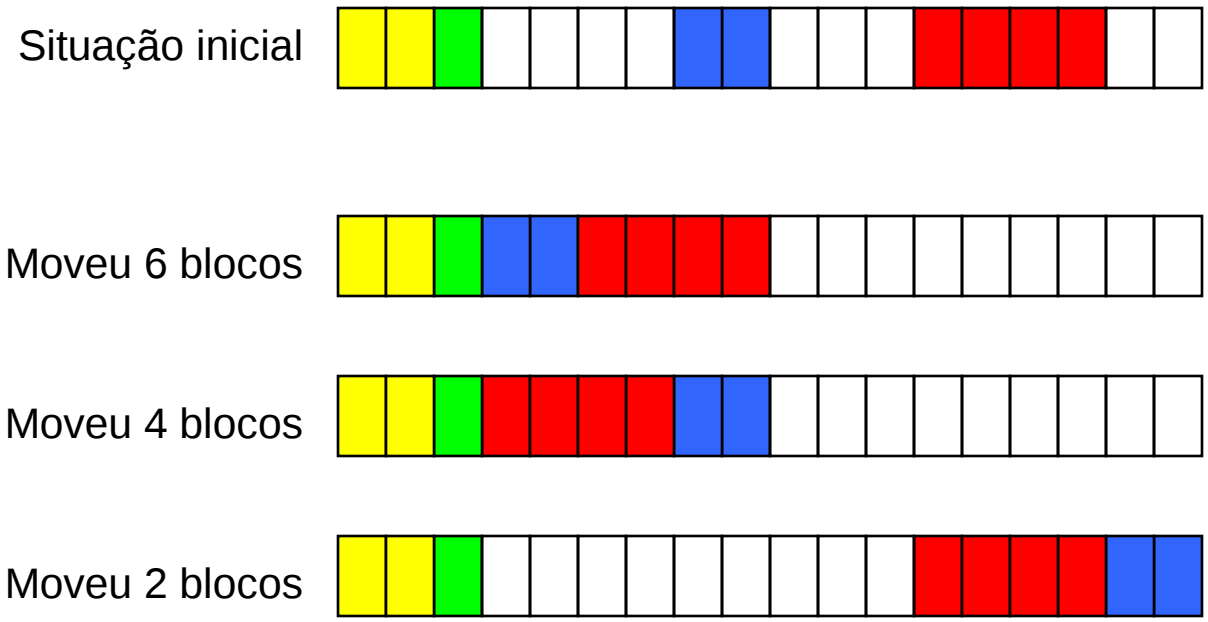


Agora, como alocar um arquivo com 4 blocos ?

Desfragmentação

- Mover arquivos para reagrupar os fragmentos em espaços maiores
- Visa permitir alocar arquivos maiores
- Deve ser feita periodicamente
- Uso de algoritmos para minimizar movimentação de arquivos (rapidez)

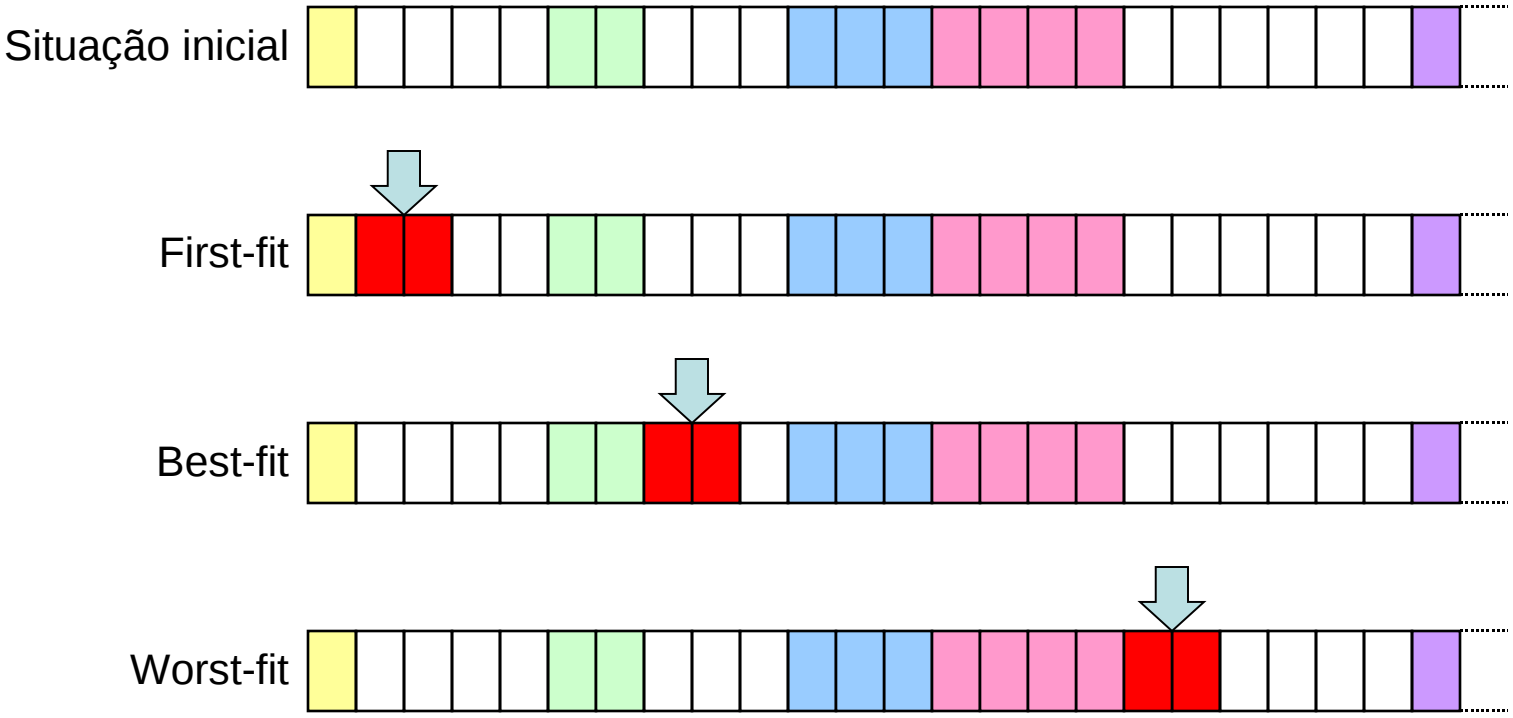
Estratégias de desfragmentação



Estratégias de alocação

- First-fit: usar o **primeiro** espaço livre
  - maior rapidez de alocação
  - pouca preocupação com fragmentos
- Best-fit: usar o **menor** espaço livre
  - usar o melhor possível os espaços em disco
  - fragmentos residuais são pequenos
- Worst-fit: usar o **maior** espaço livre
  - fragmentos residuais são maiores (mais úteis)

Alocando um arquivo c/ 2 blocos

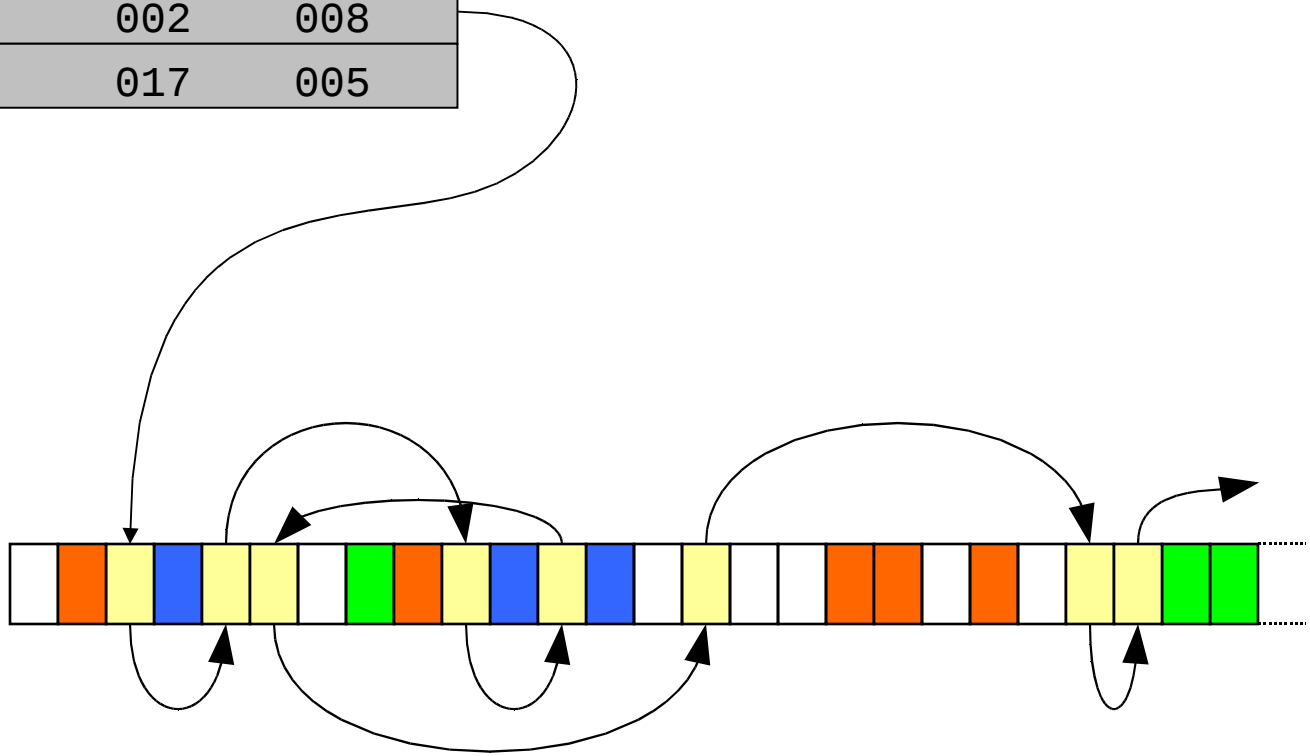


Alocação encadeada

- Os arquivos são armazenados como listas de blocos
  - cada bloco aponta para o próximo
  - diretório aponta para o bloco inicial
  - os blocos podem estar espalhados
- Base de funcionamento da FAT
  - sistema de arquivos Windows

Alocação encadeada

arquivo	inicio	#blocos
readme.txt	010	003
prova.doc	002	008
Aula.pdf	017	005



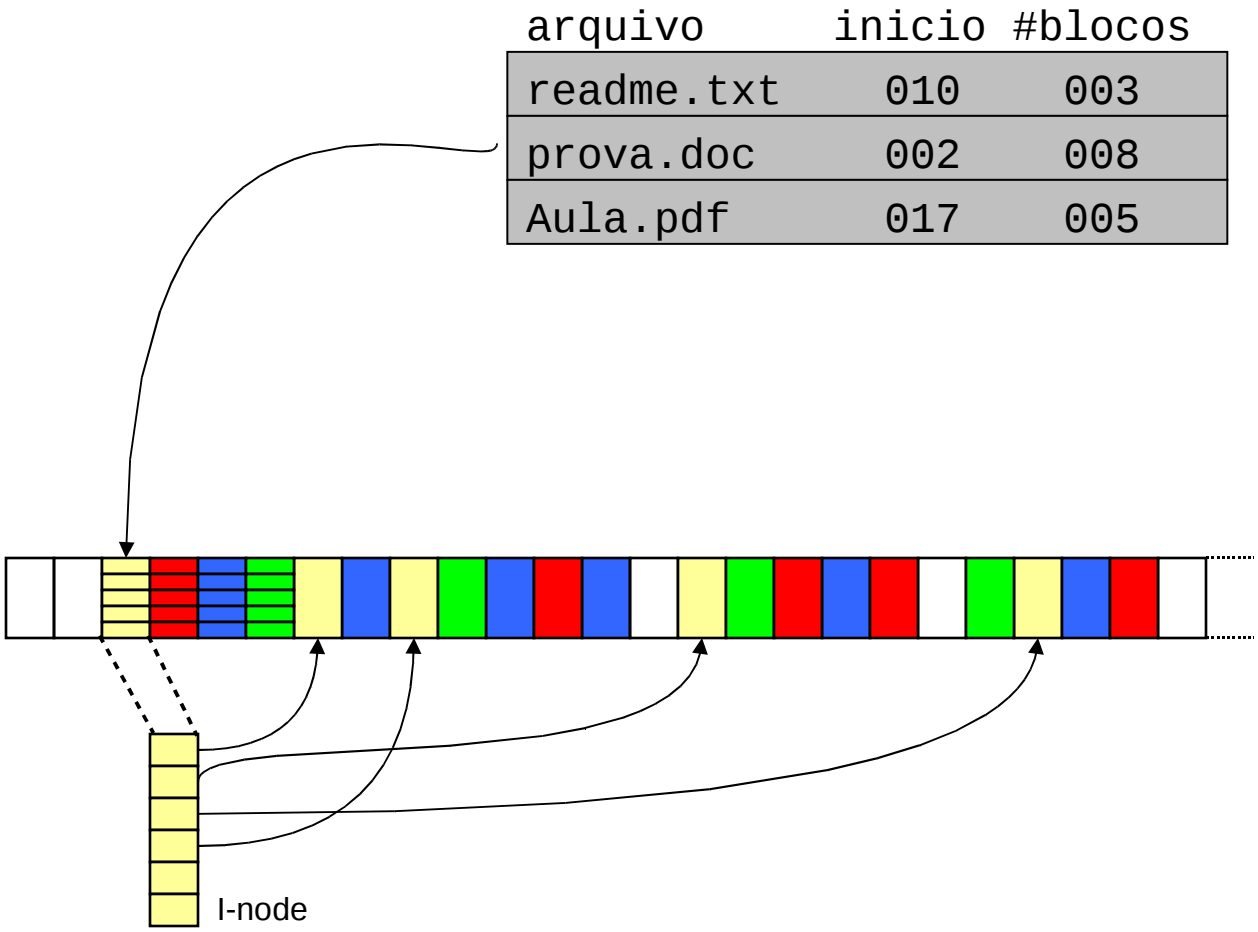
Alocação encadeada

- Vantagens
  - não há fragmentação externa
  - todo o disco pode ser usado
  - tamanho dos arquivos pode ser mudado facilmente
- Desvantagens
  - acesso aleatório é mais demorado
  - maior fragilidade em caso de problemas

Alocação indexada

- Baseada em tabelas de blocos
  - um bloco especial guarda a tabela de blocos do arquivo: index-node (i-node)
  - diretório aponta para os i-nodes
  - blocos podem estar espalhados
- Base de funcionamento do UNIX

Alocação indexada



Alocação indexada

- Vantagens
  - não há fragmentação externa
  - todo o disco pode ser usado
  - acesso rápido
  - robustez em caso de problemas
- Desvantagens
  - gerência mais complexa
  - espaço em disco perdido com os i-nodes

Fragmentação interna

- Arquivos são alocados em blocos:
  - Os blocos têm tamanho fixo.
  - Entre 512 bytes e 8 Kbytes.
  - Um bloco não pode ser alocado parcialmente.
- Se usarmos blocos de 4096 bytes:
  - um arquivo de 5700 bytes ocupará 2 blocos.
  - 2492 bytes serão perdidos no último bloco.
- Em média, perde-se 1/2 bloco por arquivo.

Tamanho dos blocos

- A escolha do tamanho dos blocos é importante para a eficiência do sistema.

- Blocos pequenos:
  - menor perda por fragmentação interna
  - mais blocos por arquivo: maior custo de gerência
- Blocos grandes:
  - maior perda por fragmentação interna
  - menos blocos por arquivo: menor custo de gerência