Sistemas de Arquivos: Implementação

Sistemas Operacionais

Prof. Pedro Ramos
pramos.costar@gmail.com

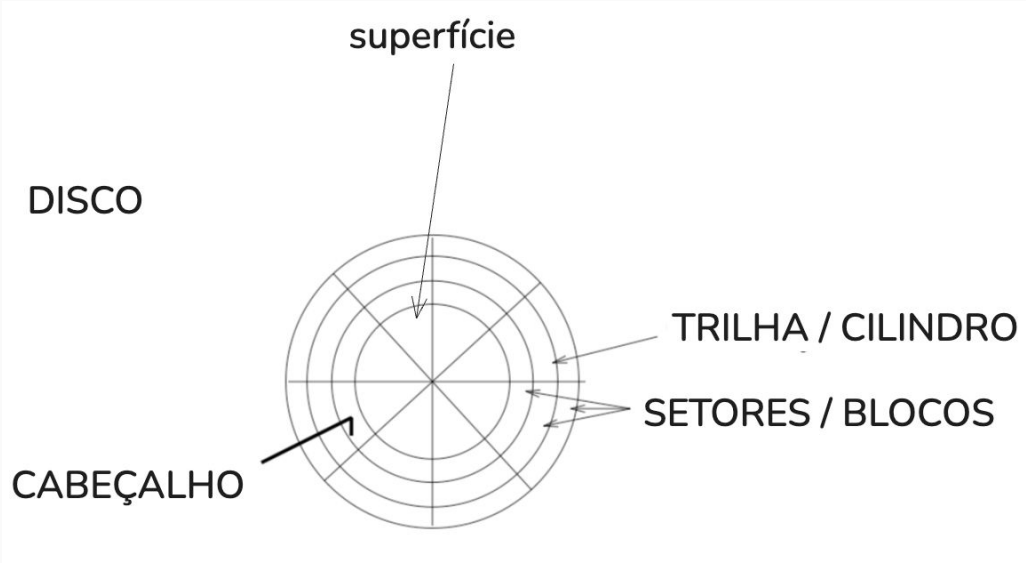
Pontifícia Universidade Católica de Minas Gerais
ICEI - Departamento de Ciência da Computação

COMO DISCOS FUNCIONAM

A superfície do disco é circular e revestida com um material **magnético**.

O disco está **sempre girando** (como um CD).

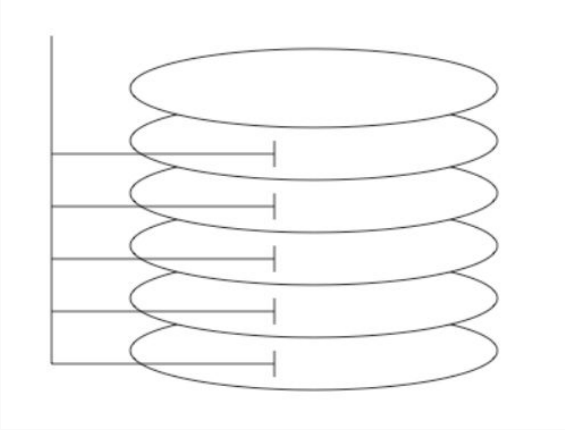
- Trilhas são anéis concêntricos no disco, com bits organizados em série ao longo das trilhas.



- Cada trilha é dividida em setores ou blocos, que são a unidade mínima de transferência do disco, MESMO ao ler 1 só bit.

COMO DISCOS FUNCIONAM

- CDs são unidades, mas discos vêm organizados em **packs de discos**, que consistem em uma pilha de **pratos.**
- **Packs de discos** utilizam ambos os lados dos pratos, exceto nas extremidades.
- O **pente (leitor/escritor)** possui dois conjuntos de cabeças de leitura/gravação na extremidade de cada braço.
- **Cilindros** são setores correspondentes em cada superfície.
- As operações no disco são realizadas em termos de **coordenadas radiais**:
  - Movendo o braço para a trilha correta e aguardando o disco girar até posicionar o setor sob a cabeça.
  - Selecionando e transferindo o setor correto conforme ele passa girando.



SOBRECARGAS (OVERHEADS) DO DISCO

- **Overhead**: tempo que a CPU leva para iniciar uma operação de disco.

- **Latência**: tempo necessário para iniciar a transferência de 1 byte do disco para a memória.
  - **Tempo de busca (*seek time*)**: tempo para posicionar a cabeça de leitura/escrita no cilindro correto.
  - **Tempo de rotação**: tempo necessário para que o setor correto gire até estar sob a cabeça.
- **Largura de banda**: uma vez iniciada a transferência, é a taxa de transferência de I/O.

ORGANIZAÇÃO DO ARQUIVO NO DISCO

COMO FICA O ARQUIVO NO DISCO?

As informações necessárias:

- **fileID 0, Bloco 0** → Prato 0, cilindro 0, setor 0
- **fileID 0, Bloco 1** → Prato 4, cilindro 3, setor 8

  ...

Principais questões de desempenho:

1. Precisamos suportar acesso **sequencial e aleatório.**
2. Qual é a **melhor estrutura de dados** para armazenar as informações de localização dos arquivos?
3. **Como organizar** os arquivos no disco físico?

ORGANIZAÇÃO DO ARQUIVO NO DISCO

Estruturas de Dados no Disco

- A estrutura usada para descrever onde o arquivo está no disco e os atributos do arquivo é o **descritor de arquivo (FileDesc)**. Os descritores de arquivo também precisam ser armazenados no disco, assim como os arquivos.

- A maioria dos sistemas segue o seguinte perfil:
  1. A maioria dos arquivos é pequena.
  2. A maior parte do espaço do disco é ocupada por arquivos grandes.
  3. Operações de I/O são realizadas tanto em arquivos pequenos quanto grandes.

=> O custo por arquivo precisa ser baixo, mas arquivos grandes também devem ter um bom desempenho.

ALOCAÇÃO CONTÍGUA

- O mais simples: O **sistema operacional (SO)** mantém uma lista ordenada de blocos livres no disco.
- O SO aloca um bloco contíguo de blocos livres ao criar um arquivo.
- É necessário armazenar apenas a **localização inicial** e o **tamanho** no **descritor de arquivo**.

Vantagens

- Simplicidade.
- Tempo de acesso? Quantidade de buscas (seeks)?

Desvantagens

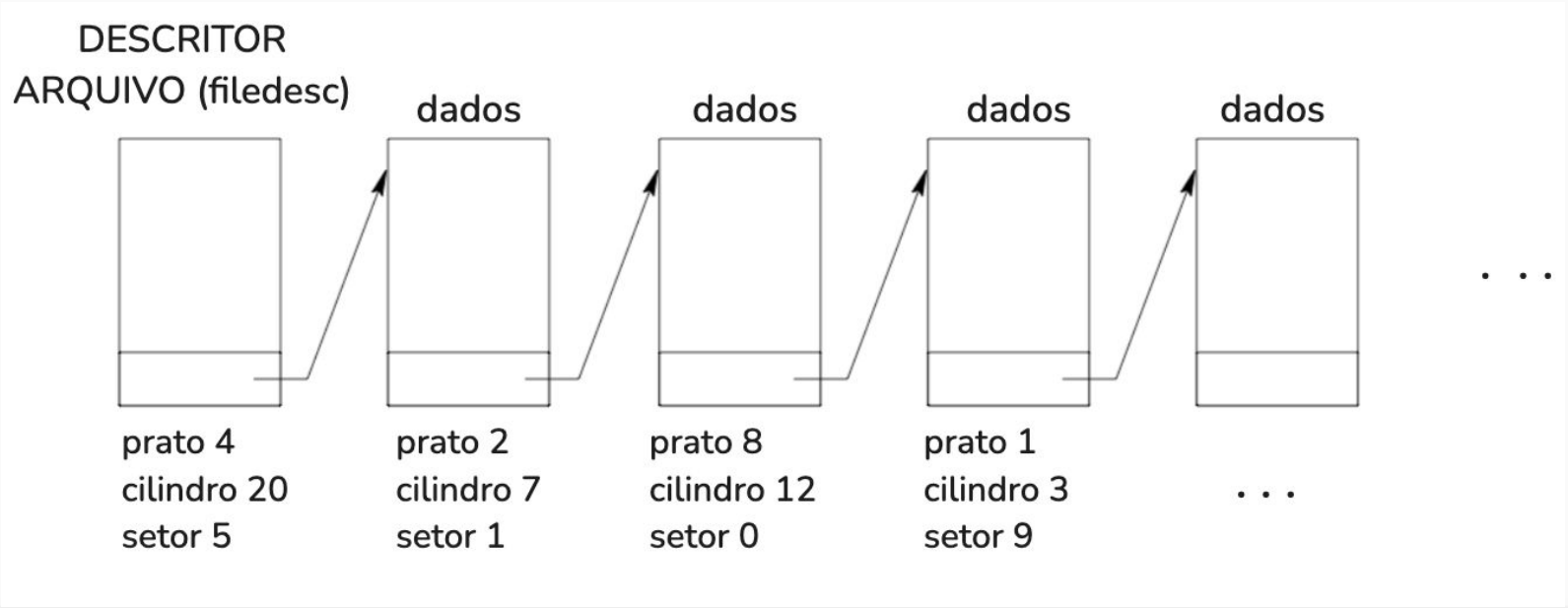
- Alteração no tamanho dos arquivos.
- Fragmentação? Gerenciamento do disco?

Exemplos

- IBM OS/360, discos de gravação única, primeiros computadores pessoais.

ARQUIVOS ENCADEADOS

- Manter uma lista de todos os setores/blocos livres.
- No **descritor de arquivo**, armazenar um ponteiro para o primeiro setor/bloco.
- Em cada setor, armazenar um ponteiro para o próximo setor.



ARQUIVOS ENCADEADOS

Vantagens

- Fragmentação? Sem fragmentação: os blocos podem estar em qualquer lugar no disco.
- Alterações no tamanho do arquivo? Fácil de gerenciar: atualizar os ponteiros, adicionar ou remover blocos.
- Suporta de forma eficiente qual tipo de acesso?

  Acesso sequencial, pois os blocos são lidos em ordem.

Desvantagens

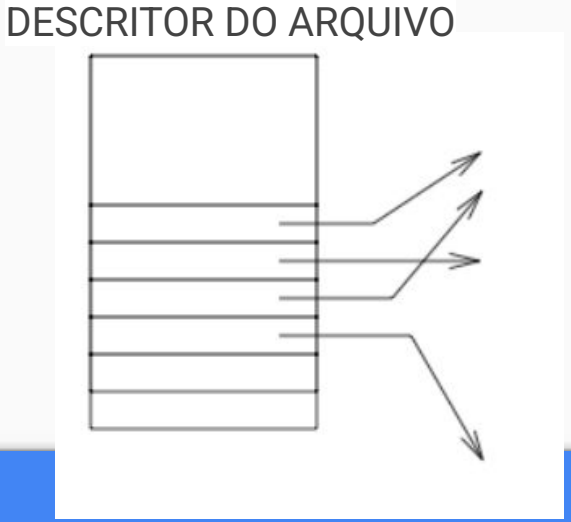
- Não suporta bem qual tipo de acesso? **Acesso aleatório**, pois é necessário seguir os ponteiros para localizar o bloco desejado.
- Quantidade de buscas (seeks)? Pode ser alta devido à dispersão dos blocos no disco.

Exemplo

- **MS-DOS**

ARQUIVOS INDEXADOS

- O **SO** mantém um array de ponteiros para blocos para cada arquivo.
- O usuário ou o SO deve declarar o tamanho máximo do arquivo no momento de sua criação.
- O SO aloca um array para armazenar os ponteiros de todos os blocos ao criar o arquivo, mas aloca os blocos apenas sob demanda.
- O SO preenche os ponteiros à medida que aloca os blocos.



ARQUIVOS INDEXADOS

Vantagens

- Pouco espaço desperdiçado.
- Tanto o acesso **sequencial** quanto o **aleatório** são fáceis de implementar.

Desvantagens

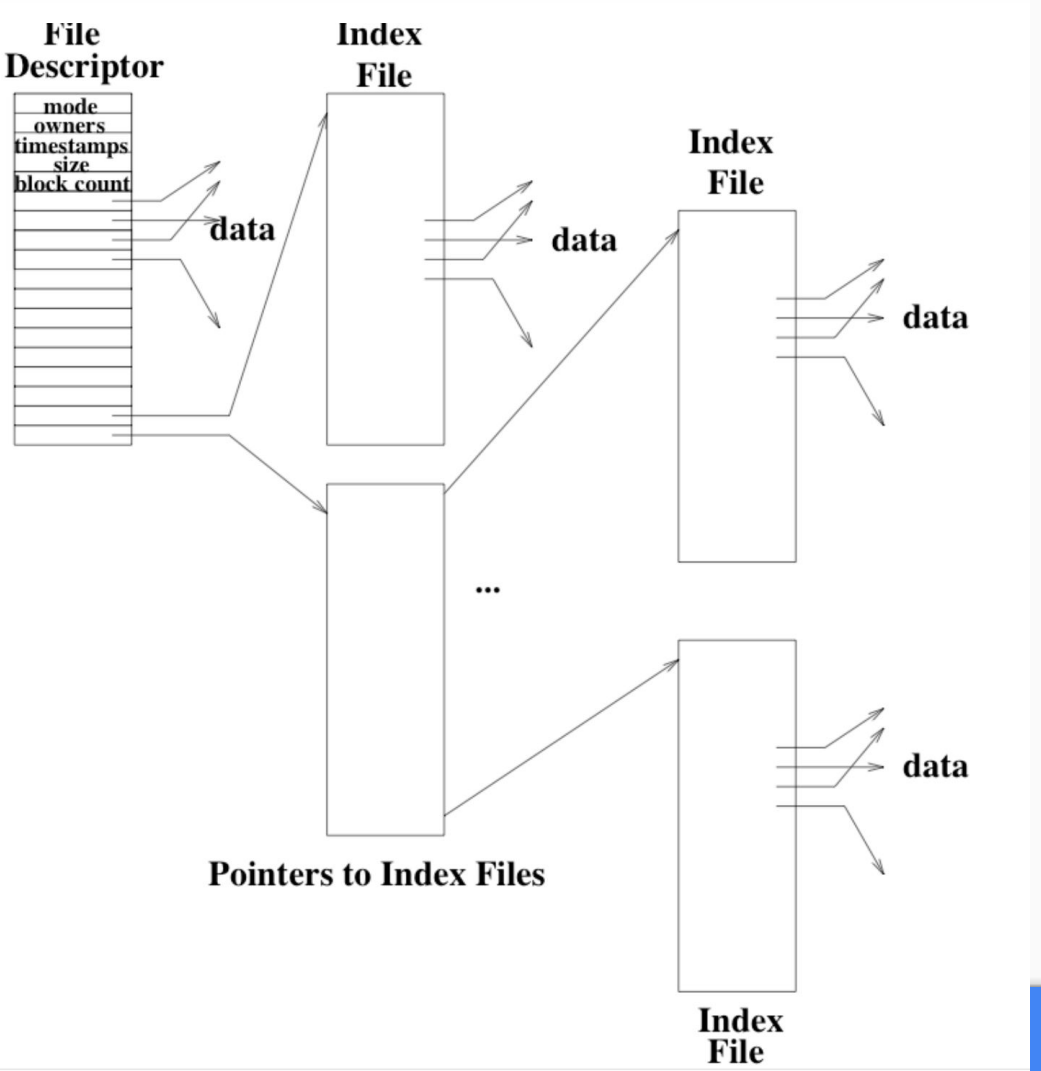
- 1 ponteiro para cada bloco (aumento no tamanho do arquivo desc)
- Define um tamanho máximo para o arquivo.
- Muitos **seeks** devido à dispersão dos dados, já que os blocos não são contíguos.

Exemplo

- **Nachos**

ARQUIVOS INDEXADOS EM MÚLTIPLOS NÍVEIS

- Cada **descritor de arquivo** contém 14 ponteiros para blocos.
- Os primeiros **12 ponteiros** apontam diretamente para blocos de dados.
- O **13º ponteiro** aponta para um bloco que contém **1024 ponteiros** para mais 1024 blocos de dados (**uma indireção**).
- O **14º ponteiro** aponta para um bloco de ponteiros que referenciam blocos indiretos (**duas indireções**).



ARQUIVOS INDEXADOS EM MÚLTIPLOS NÍVEIS - BSD/UNIX 4.3

Vantagens

- Implementação simples.
- Suporta crescimento incremental do arquivo.
- Arquivos pequenos? **Posso usar o espaço do arquivo descritor pra guardá-los sem precisar de *seek()*'s extras.**

Desvantagens

- Acesso indireto é ineficiente para acesso aleatório em arquivos muito grandes.
- Muitos **seeks em arquivos grandes**, pois os dados não estão em blocos contíguos.

Questões

- O tamanho do arquivo é limitado? *Sim, devido à quantidade finita de ponteiros.*
- O que o SO poderia fazer para obter acesso mais contíguo e reduzir o número de **seeks**?
  - **Alocar blocos adjacentes sempre que possível.**

GERENCIAMENTO DE ESPAÇO LIVRE

- É necessário manter uma **lista de espaço livre** para rastrear quais blocos do disco estão disponíveis *(assim como fazemos com a memória principal)*.

- Deve ser possível encontrar espaço livre rapidamente e liberar espaço com eficiência. Para isso, utiliza-se um **bitmap**:
  - O bitmap possui **um bit** para cada bloco do disco.
  - Se o bit é **1**, o bloco está livre. Se o bit é **0**, o bloco está alocado.

GERENCIAMENTO DE ESPAÇO LIVRE

Vantagens do Bitmap

- É possível verificar rapidamente se há alguma página livre nas próximas 32, comparando a palavra com 0:
  - Se for **0**, todas as páginas estão em uso.
  - Caso contrário, operações bit a bit podem identificar rapidamente um bloco vazio (exemplo: 11000010010001111111 0...).

- Marcar um bloco como liberado é simples, já que o número do bloco pode ser usado como índice no bitmap para ajustar o bit correspondente.

GERENCIAMENTO DE ESPAÇO LIVRE

Problema

- O **bitmap** pode ser muito grande para ser mantido na memória principal em discos de grande capacidade.
  - Exemplo: Um disco de **2 GB** com setores de **512 bytes** requer um bitmap com **4.000.000 entradas** (500.000 bytes).
- Se a maior parte do disco estiver em uso, encontrar blocos livres com o bitmap pode ser caro.

Solução Alternativa

- **Encadear os blocos livres**:
  - O cabeçalho da lista é mantido em memória no kernel. Cada bloco contém um ponteiro para o próximo bloco livre.

Questões de Desempenho

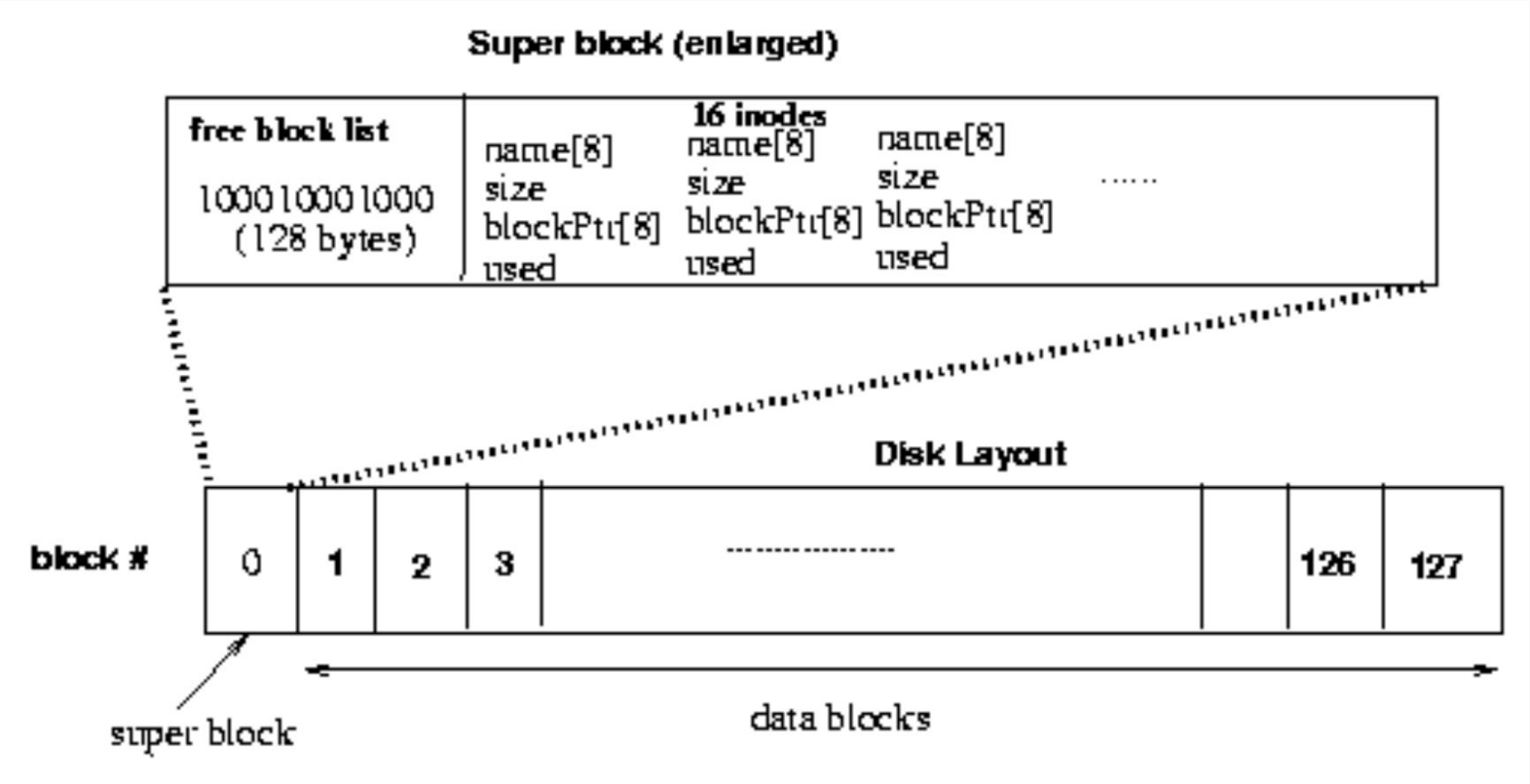
- Qual o custo para **alocar um bloco**?
- Qual o custo para **liberar um bloco**?
- Qual o custo para **alocar blocos consecutivos**?

SUPERBLOCO

- Nos sistemas de arquivos Unix, o **superbloco** é um bloco especial do disco usado para armazenar **metadados**.
- Os **superblocos** são armazenados em locais conhecidos, como no **trilho 0** (*track 0*).
- O **superbloco** armazena os **inodes** de forma sequencial.
- Os **inodes** possuem tamanho fixo e são **pré-alocados**.
  - Localização do inode i:

$$\text{número do bloco} = \frac{\text{tamanho do inode} \times i}{\text{tamanho do bloco}}$$

SUPERBLOCO


Super block (enlarged)
free block list
1000 1000 1000
(128 bytes)
16 inodes
name[8]
size
blockPtr[8]
used
name[8]
size
blockPtr[8]
used
name[8]
size
blockPtr[8]
used
.....
Disk Layout
block #
0
1
2
3
126
127
super block
data blocks

RESUMO

- Muitas das preocupações e implementações dos sistemas de arquivos são semelhantes às das implementações de memória virtual.
  - **Alocação Contígua**:
    - É simples, mas sofre com **fragmentação externa**, necessidade de **compactação** e de mover arquivos à medida que crescem.
  - **Alocação Indexada**:
    - É muito similar às **tabelas de páginas**, onde uma tabela mapeia blocos lógicos de arquivos para blocos físicos no disco.
  - **Gerenciamento de Espaço Livre**:
    - Pode ser feito usando um **bitmap** ou uma **lista encadeada.**

PERGUNTAS?

REFERÊNCIAS

- **TANENBAUM, Andrew.** Sistemas operacionais modernos.
- **SILBERSCHATZ, Abraham et al.** Fundamentos de sistemas operacionais: princípios básicos.
- **MACHADO, Francis; MAIA, Luiz Paulo.** Arquitetura de Sistemas Operacionais.
- **CARISSIMI, Alexandre et al.** Sistemas operacionais.