Sistemas de Arquivos:
Implementação de Arquivos e Diretórios

Prof. Dr. Márcio Castro
marcio.castro@ufsc.br

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

1 Introdução

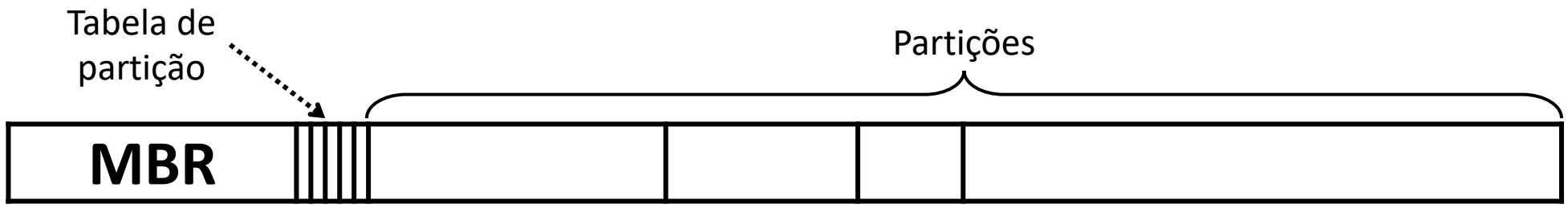
Introdução

- Os sistemas de arquivos são armazenados em **disco**
- A maioria dos discos é dividia em uma ou mais **partições**
  - Cada partição possui um **sistema de arquivos independente**
- O disco é divido em **setores**
  - O **setor 0** do disco, chamado de **Master Boot Record (MBR)**, é usado para inicializar o computador
  - O fim do MBR contém a **tabela de partição**, a qual armazena os endereços iniciais e finais de **cada partição**
  - Uma das partições na tabela é marcada como **ativa**

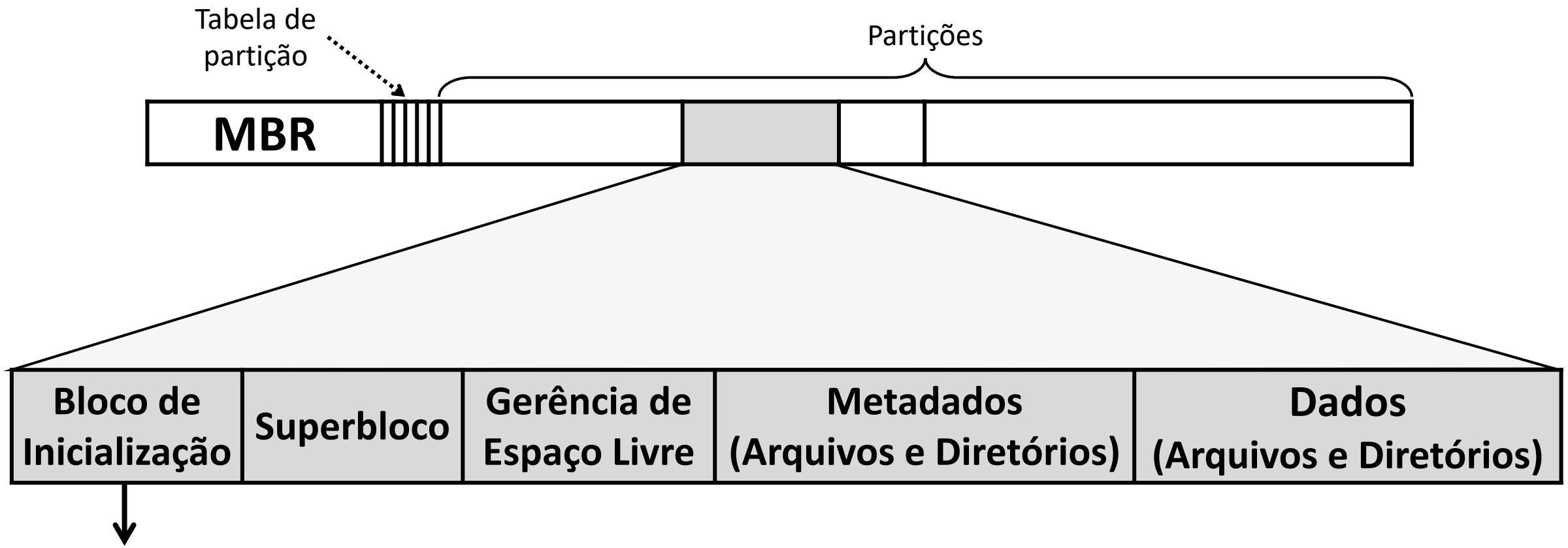
Introdução

- No momento do **boot** do computador:
  1. A **BIOS** lê e executa o **MBR**
  2. O programa do MBR localiza a **partição ativa**, lê o seu primeiro bloco (**bloco de inicialização**) e o **executa**
  3. O programa no bloco de inicialização **carrega o SO** contido naquela partição

Implementação do sistema de arquivos

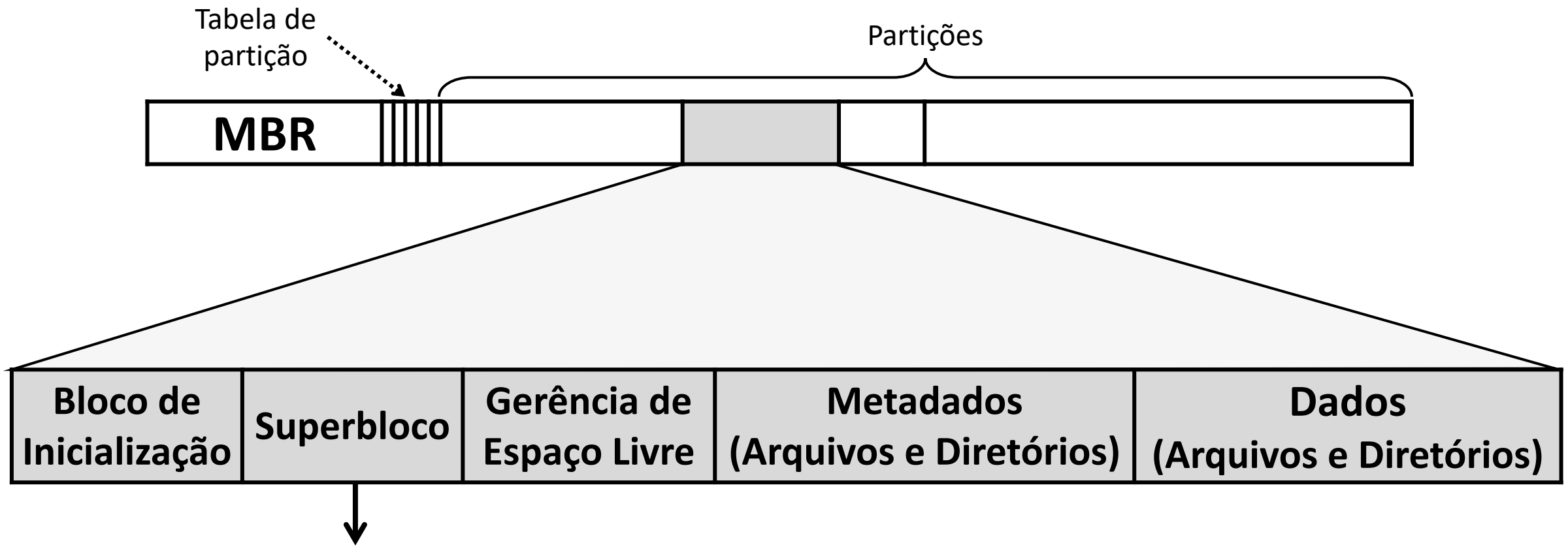


Implementação do sistema de arquivos



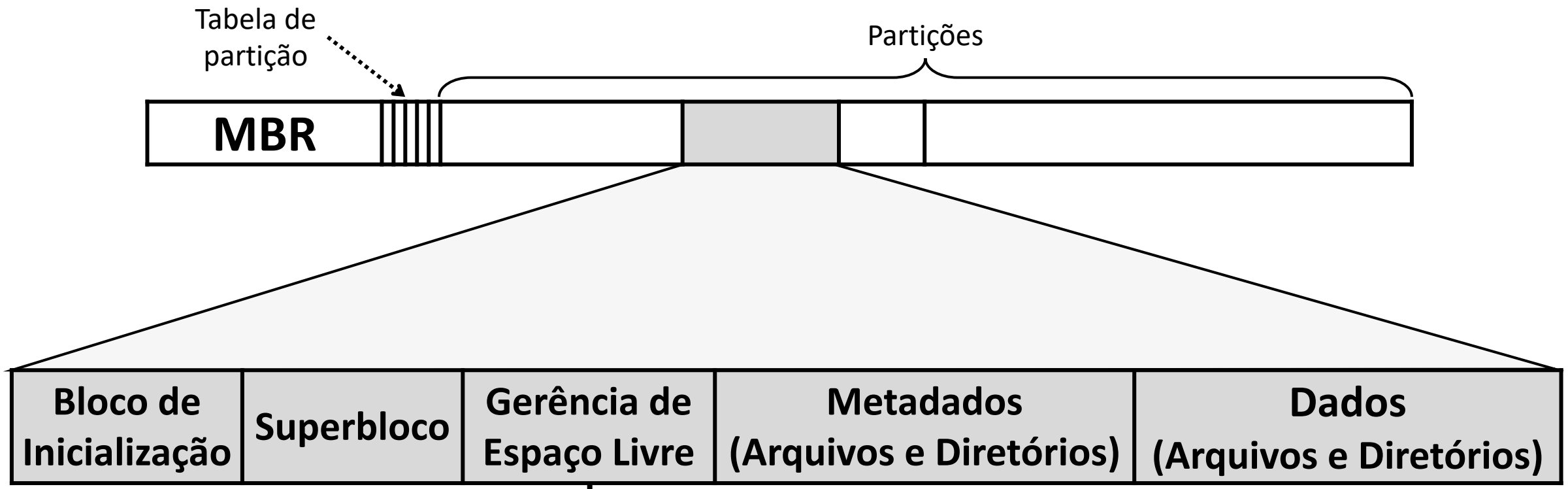
Armazena o programa responsável pelo carregamento do SO

Implementação do sistema de arquivos



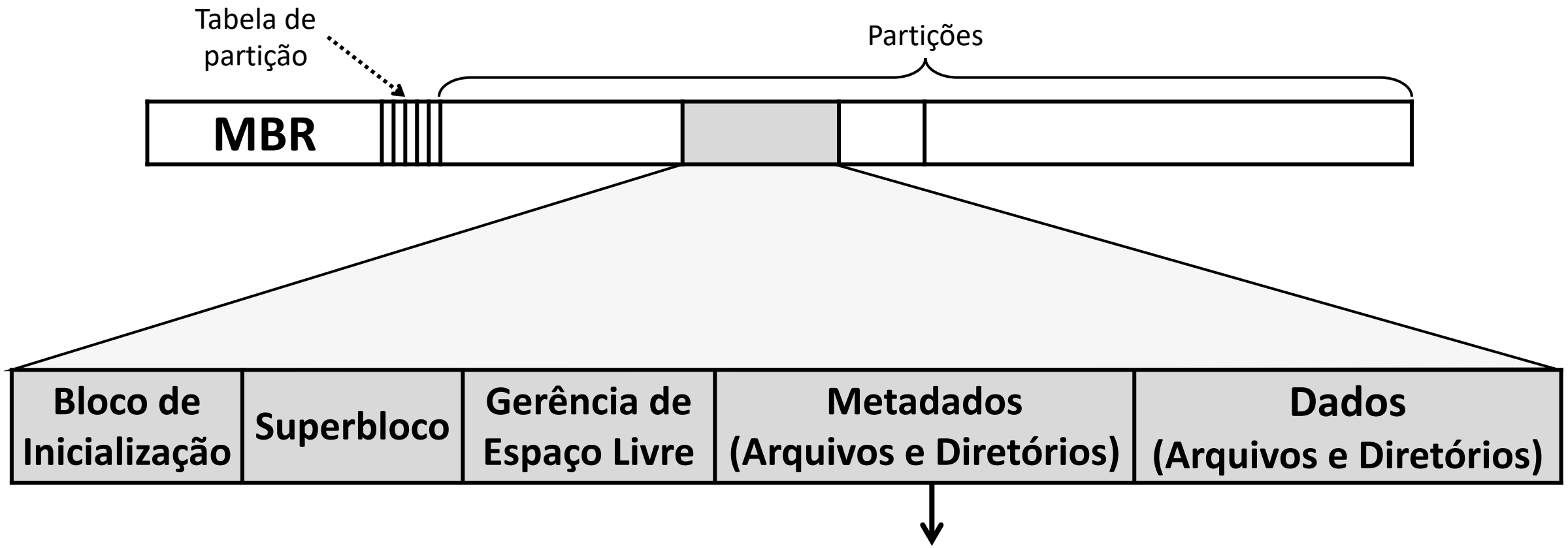
Armazena informações sobre o tipo do sistema de arquivos e o número de blocos

Implementação do sistema de arquivos



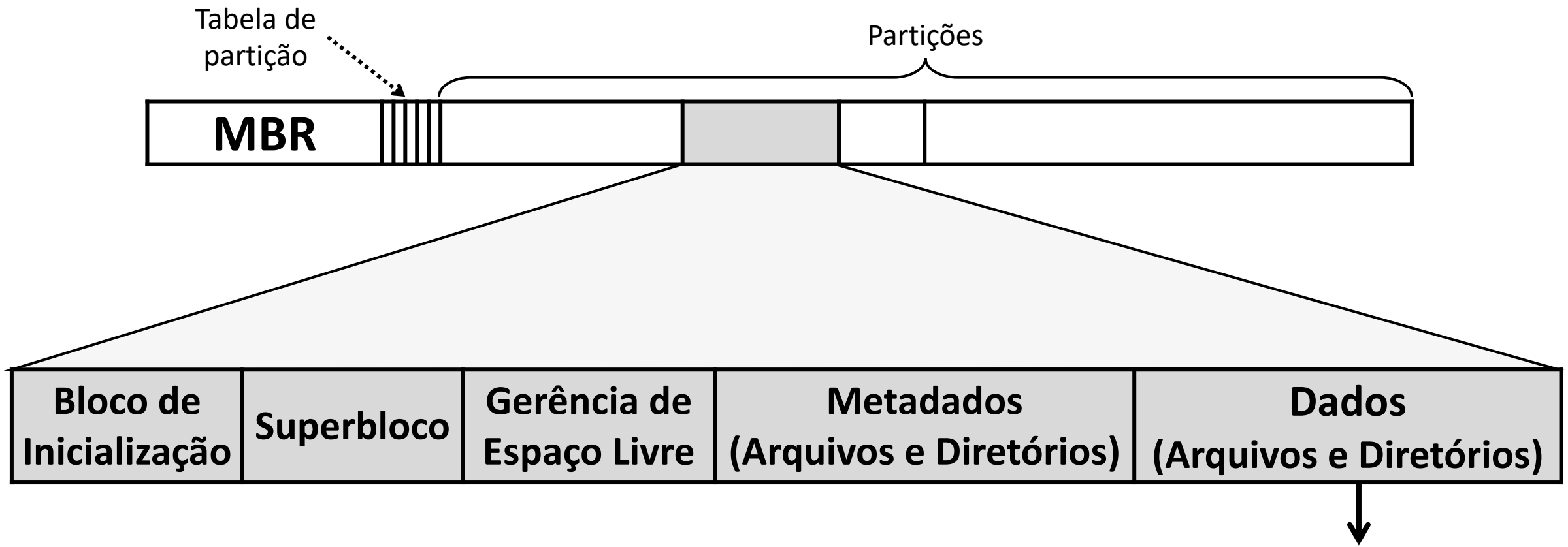
Estruturas de dados para gerenciar blocos livres de disco

Implementação do sistema de arquivos



Estruturas de dados que armazenam metadados dos arquivos e diretórios

Implementação do sistema de arquivos



Armazenamento dos dados dos arquivos e diretórios

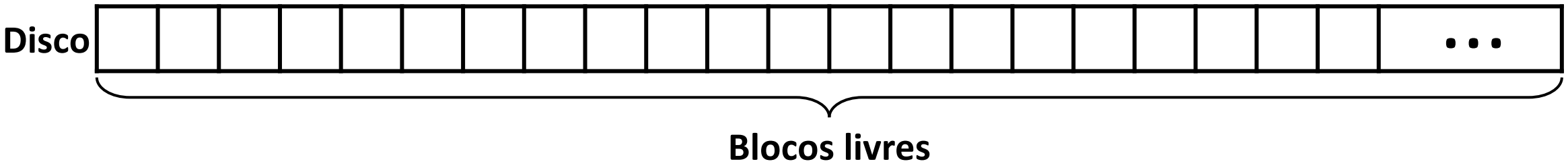
2 Implementação de arquivos

Implementação de arquivos

- O armazenamento em disco é feito em **blocos de tamanho fixo**
- O sistema de arquivos necessita **relacionar blocos do disco** com os **arquivos**
- Arquivos podem ocupar **um ou mais blocos do disco**, em função da **quantidade de dados** que armazenam
- **Estratégias para alocação de blocos do disco**
  1. Alocação contígua de blocos
  2. Alocação de blocos por lista encadeada
  3. Alocação de blocos por lista encadeada com tabela em memória
  4. Inodes

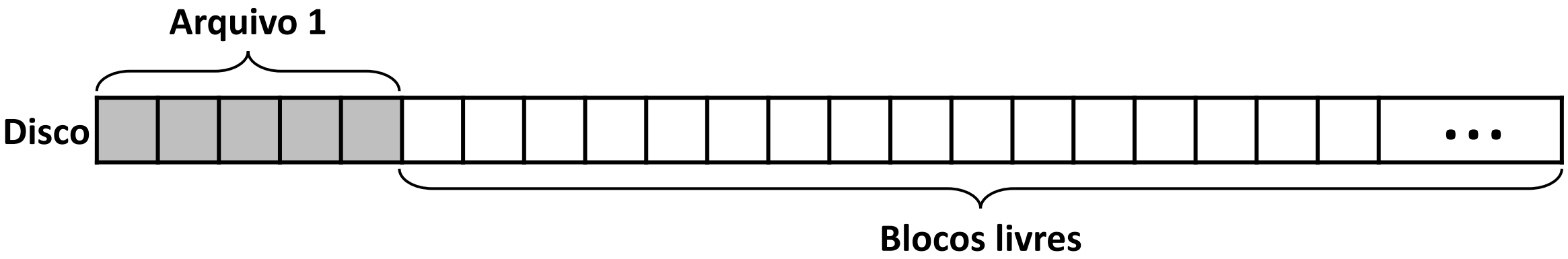
Alocação contígua de blocos

- Armazena os dados dos arquivos em **blocos contíguos do disco**



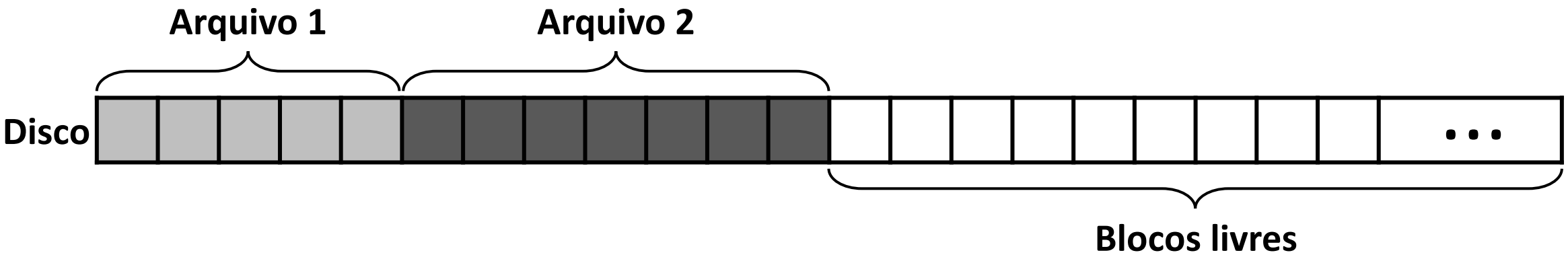
Alocação contígua de blocos

- Armazena os dados dos arquivos em **blocos contíguos do disco**



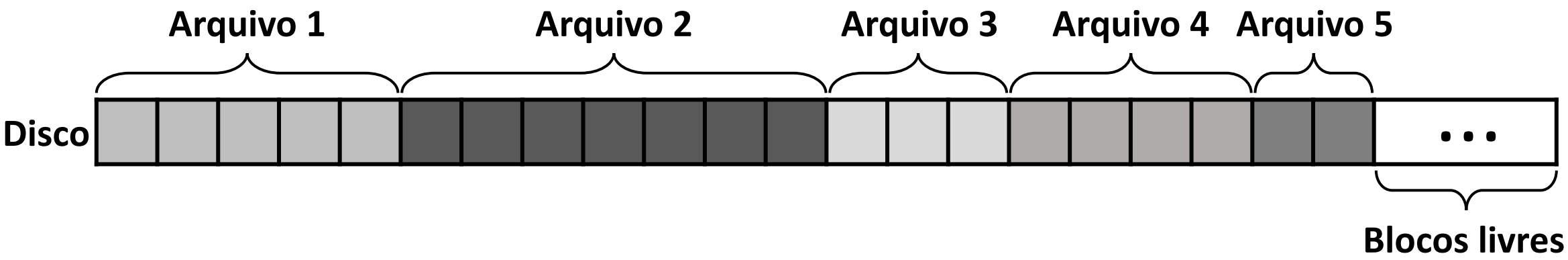
Alocação contígua de blocos

- Armazena os dados dos arquivos em **blocos contíguos do disco**



Alocação contígua de blocos

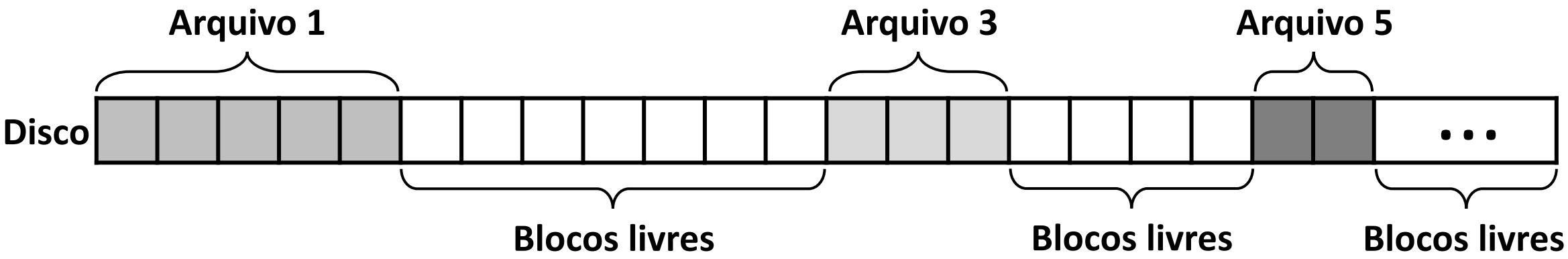
- Armazena os dados dos arquivos em **blocos contíguos do disco**



- **Vantagens**
  - **Simplicidade** de implementação
  - **Ótimo desempenho** em leituras sequenciais

Alocação contígua de blocos

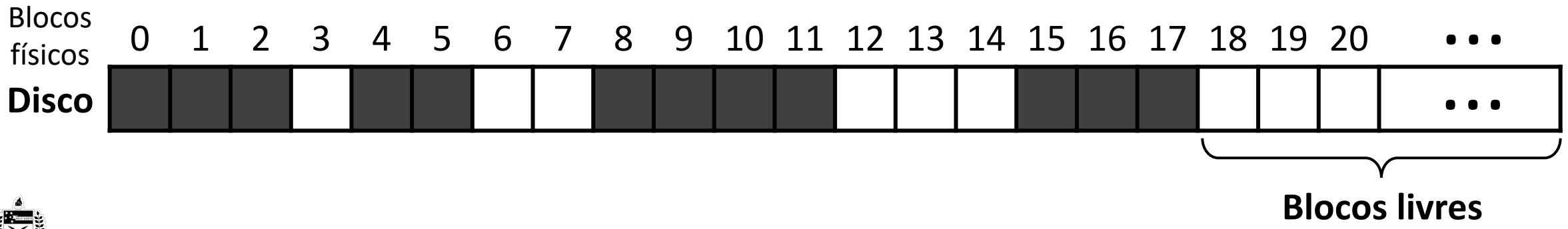
- Armazena os dados dos arquivos em **blocos contíguos do disco**



- **Vantagens**
  - **Simplicidade** de implementação
  - **Ótimo desempenho** em leituras sequenciais
- **Desvantagem**
  - **Fragmentação** do disco

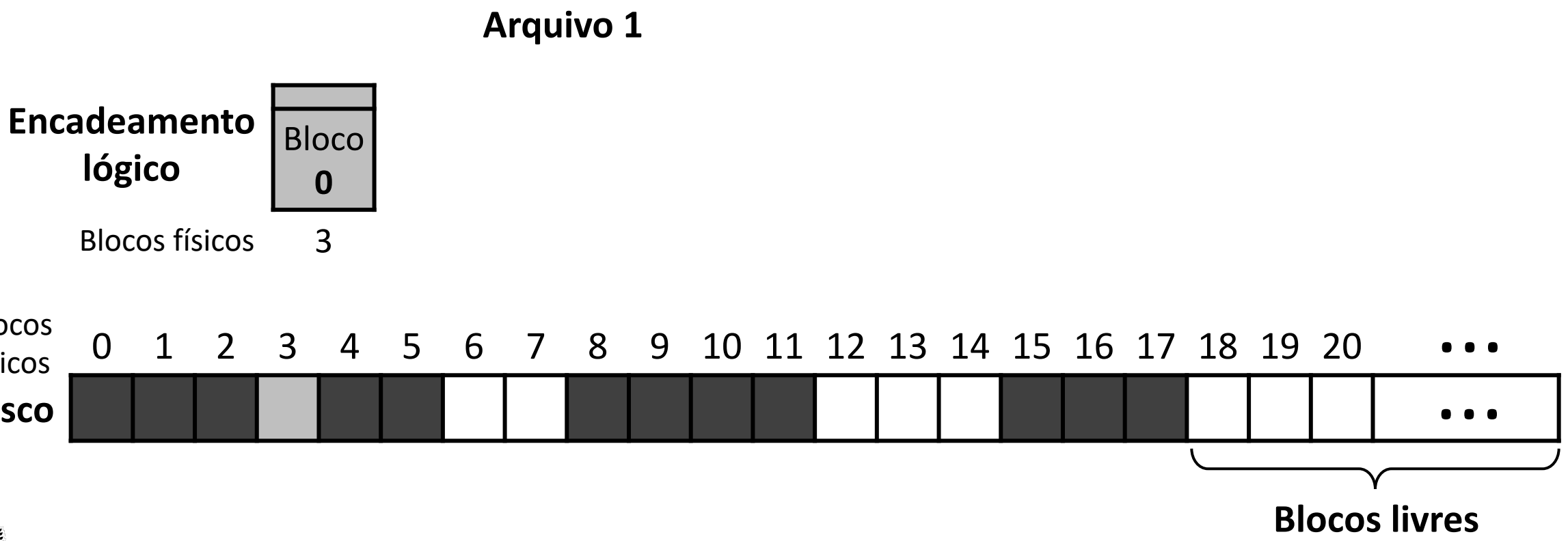
Alocação de blocos por lista encadeada

- Mantém os **blocos dos arquivos** em **listas encadeadas**
- A **primeira palavra** de cada bloco é um **ponteiro** para o próximo bloco



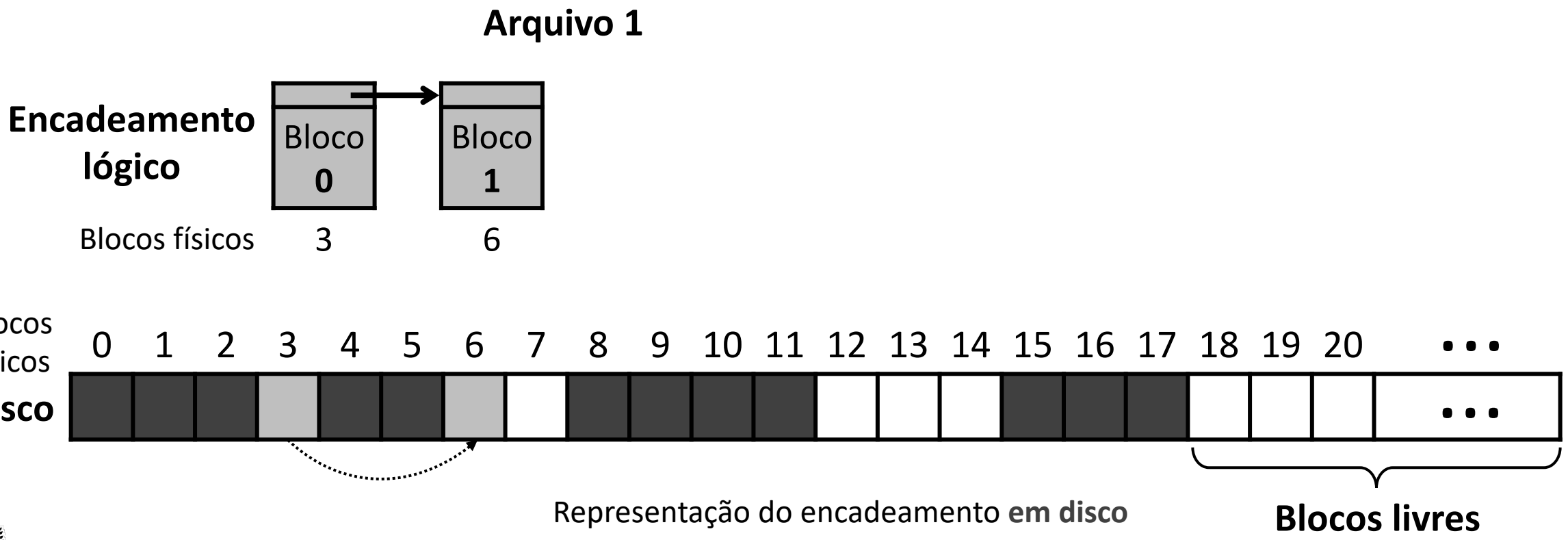
Alocação de blocos por lista encadeada

- Mantém os **blocos dos arquivos** em **listas encadeadas**
- A **primeira palavra** de cada bloco é um **ponteiro** para o próximo bloco



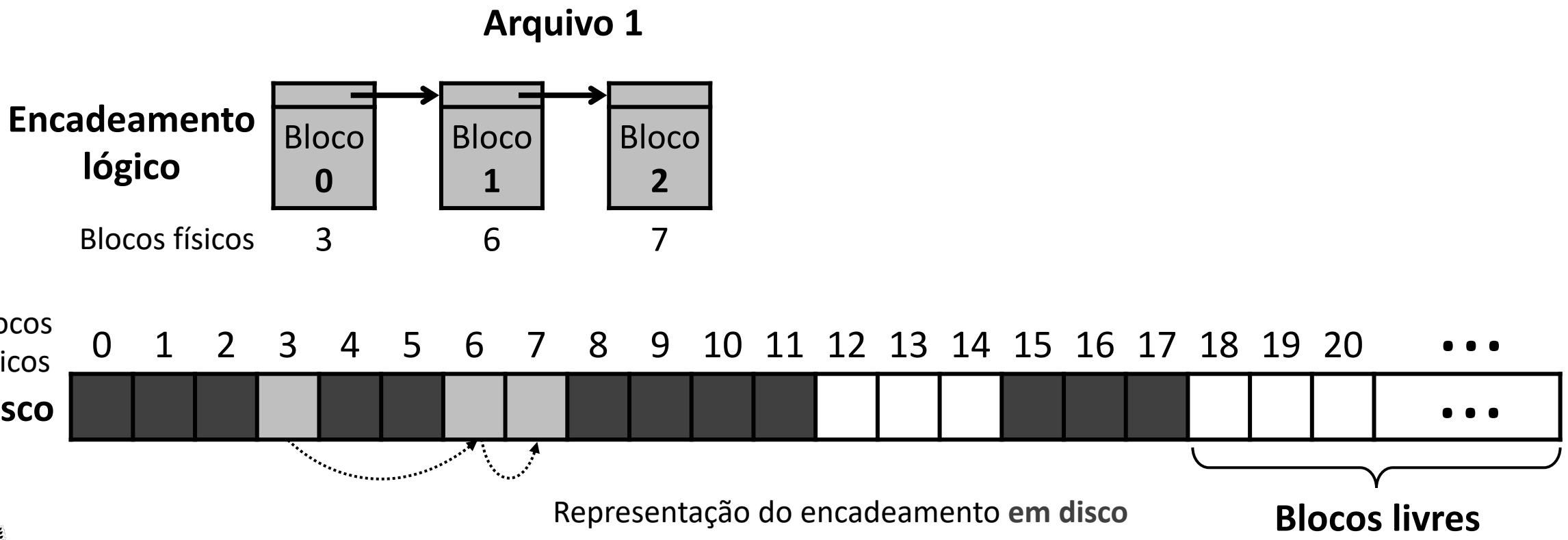
Alocação de blocos por lista encadeada

- Mantém os **blocos dos arquivos** em **listas encadeadas**
- A **primeira palavra** de cada bloco é um **ponteiro** para o próximo bloco



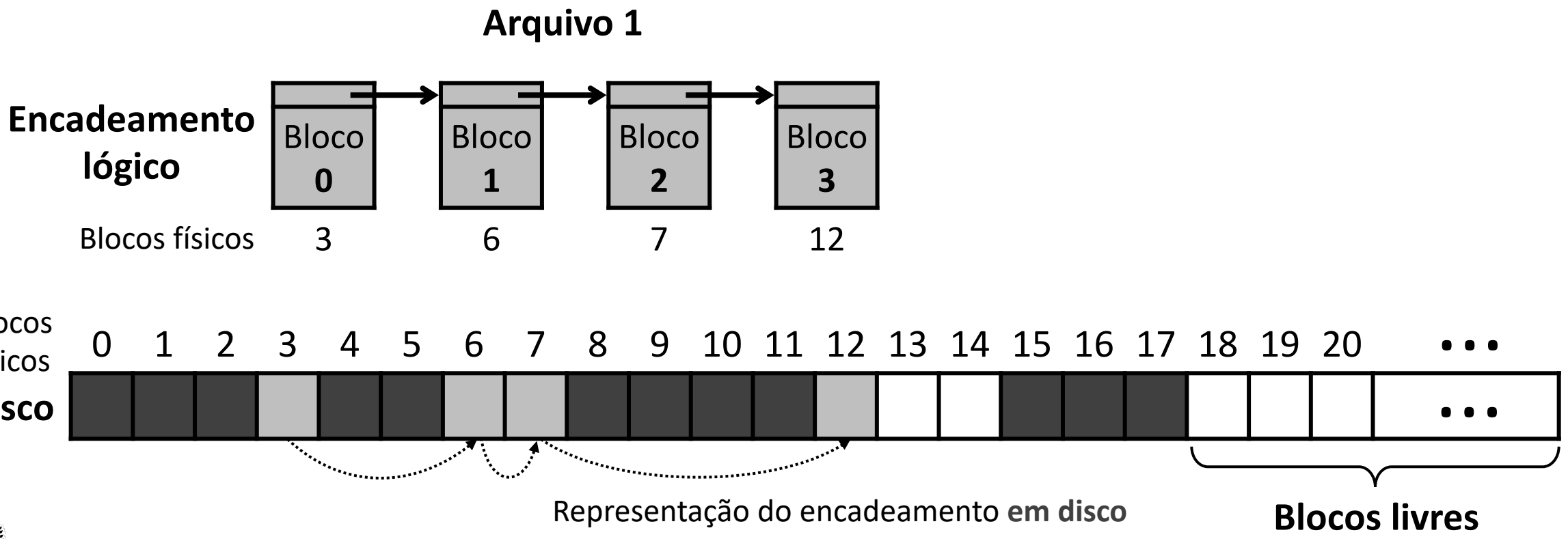
Alocação de blocos por lista encadeada

- Mantém os **blocos dos arquivos** em **listas encadeadas**
- A **primeira palavra** de cada bloco é um **ponteiro** para o próximo bloco



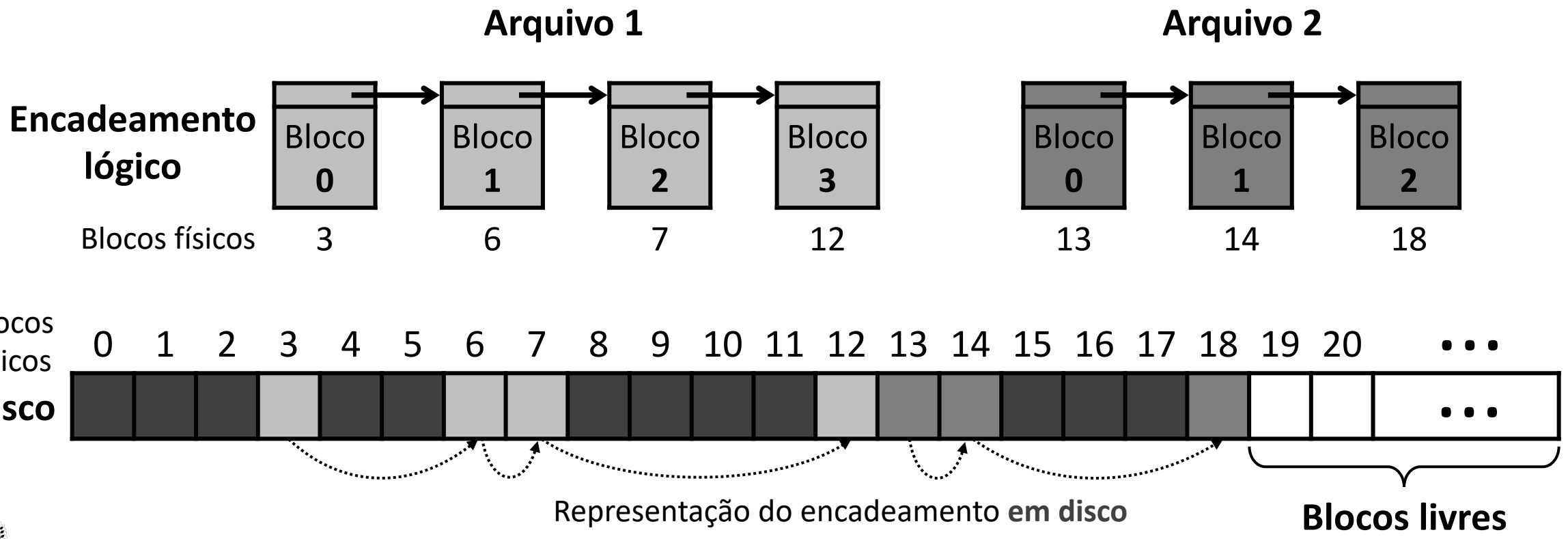
Alocação de blocos por lista encadeada

- Mantém os **blocos dos arquivos** em **listas encadeadas**
- A **primeira palavra** de cada bloco é um **ponteiro** para o próximo bloco



Alocação de blocos por lista encadeada

- Mantém os **blocos dos arquivos** em **listas encadeadas**
- A **primeira palavra** de cada bloco é um **ponteiro** para o próximo bloco



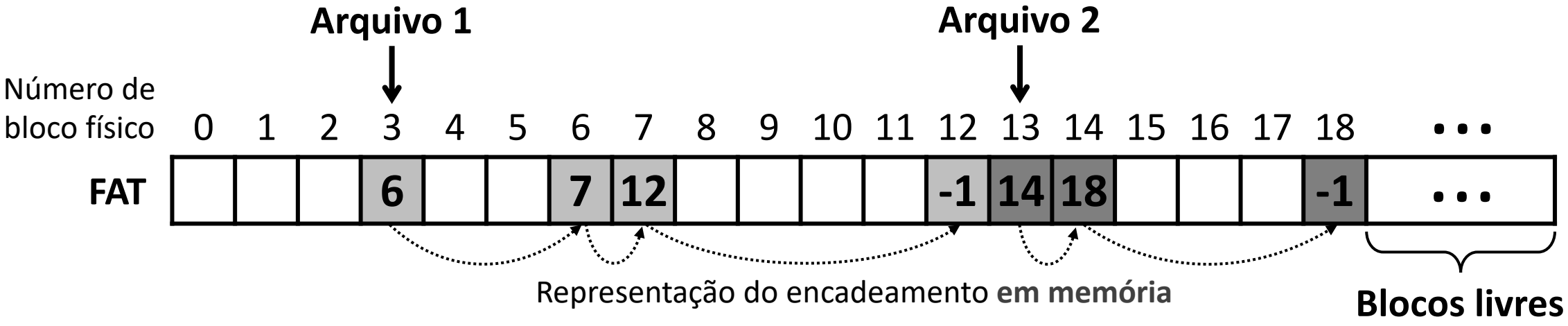
Alocação de blocos por lista encadeada

- **Vantagens**
  - Permite usar todos os blocos do disco, **evitando o problema da fragmentação**
  - **Bom desempenho** para leituras sequenciais

- **Desvantagens**
  - **Acesso aleatório** é **lento**: para chegar ao bloco n, é necessário ler $n - 1$ blocos no disco antes dele, um de cada vez
  - Parte do armazenamento de dados do bloco é **comprometido com o encadeamento de blocos**
    - Isso gera **ineficiência**, pois para ler ou escrever em um bloco inteiro de dados são necessários **acessos a 2 blocos**

Alocação de blocos por lista liga com tabela em memória

- Armazena o encadeamento de blocos em uma **tabela** denominada **File Allocation Table (FAT)**
- A FAT permanece **na memória RAM** para que o caminhamento nos blocos de um arquivo possa ser feito de maneira **eficiente**

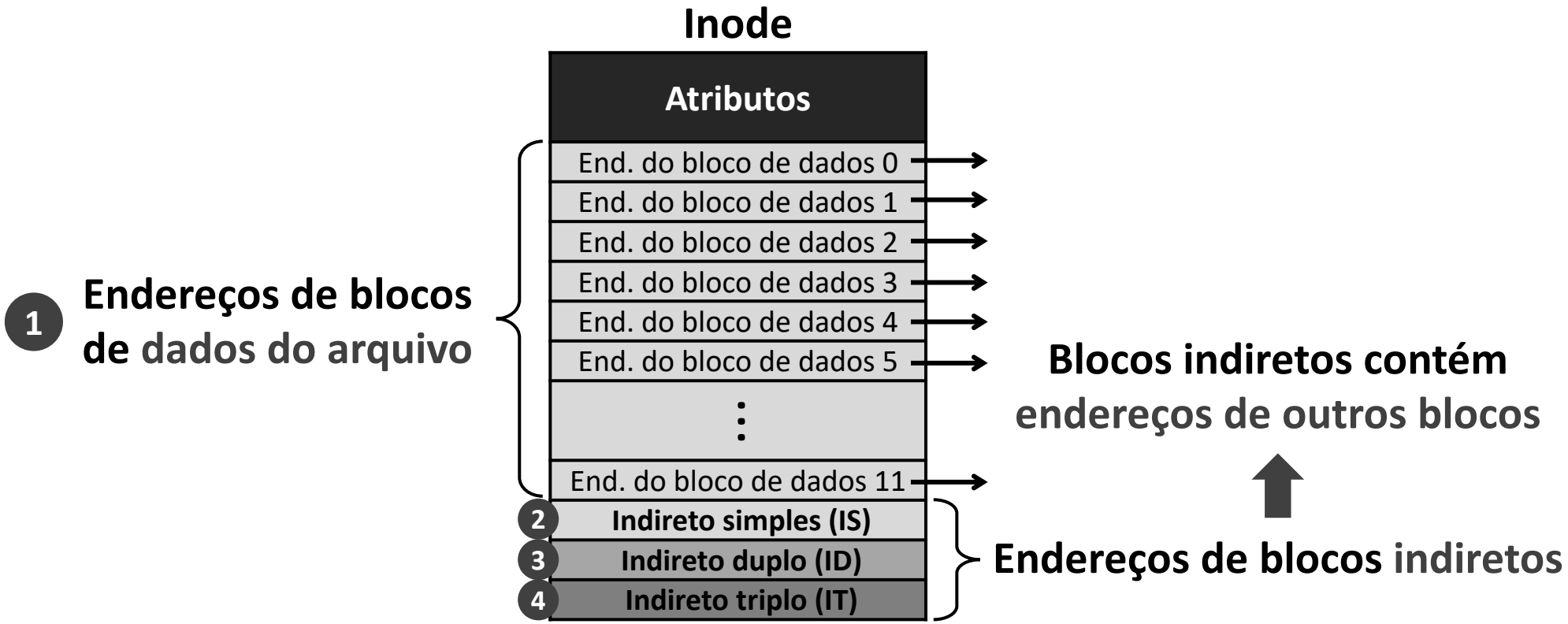


Alocação de blocos por lista liga com tabela em memória

- **Vantagens**
  - **Todo o bloco** fica disponível para dados (o encadeamento é feito na **FAT**)
  - **Acesso aleatório mais rápido**, pois não há necessidade de acessar o disco para fazer o encadeamento
- **Desvantagem**
  - A FAT precisa ser mantida na RAM e terá um **tamanho muito grande no caso de discos grandes**
  - **Exemplo:** um disco de 1 TB e blocos de disco de 1 KB, a FAT terá 1 bilhão de entradas; se cada entrada ocupa 4 bytes, a FAT ocupará 4 GB em memória

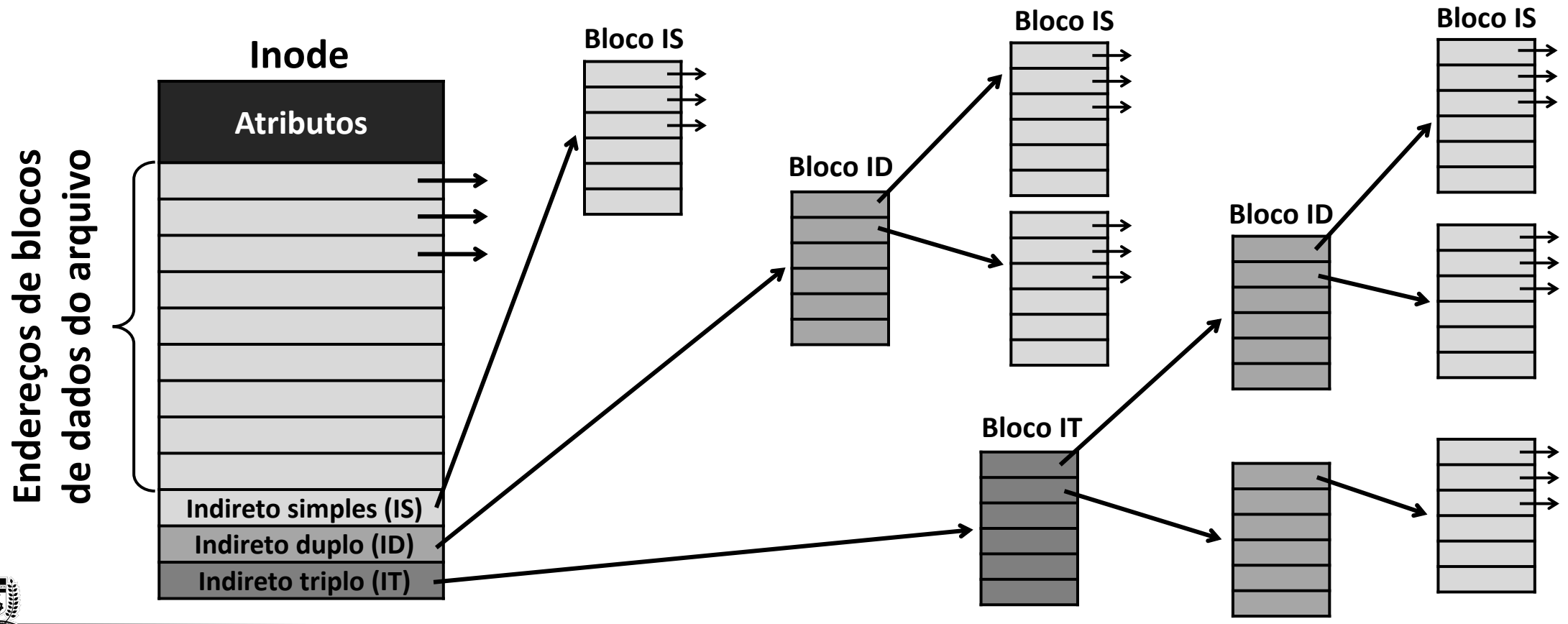
Inodes

- **Inode** é uma estrutura de dados que armazena os **atributos** do arquivo e os **endereços** de todos os seus blocos
- Inodes são organizados de uma maneira **hierárquica** de até **4 níveis**



Inodes

- **Inode** é uma estrutura de dados que armazena os **atributos** do arquivo e os **endereços** de todos os seus blocos
- Inodes são organizados de uma maneira **hierárquica** de até **4 níveis**



Inodes

- Inodes são armazenados em um **array** (**inode table**)
  - O **número de cada inode** corresponde ao seu **índice na inode table**
  - Inodes possuem diversos **atributos**

Atributo*	Descrição
`i_mode`	Define o tipo de arquivo: regular, diretório, dispositivo de caractere, dispositivo de bloco, ...
`i_uid`	ID do usuário proprietário
`i_size_lo`	32 bits menos significativos do tamanho do arquivo (em bytes)
`i_atime`	Momento em que o inode foi acessado pela última vez
`i_mtime`	Momento em que os dados do arquivo foram modificados pela última vez
`i_crtime`	Momento em que o arquivo foi criado
`i_links_count`	Número de hard links

***Lista completa em**: https://www.kernel.org/doc/html/latest/filesystems/ext4/dynamic.html

Inodes

- **Exemplos de sistema de arquivos que utilizam inodes**
  - Ext2, Ext3, Ext4, BFS, APFS, ReiserFS, ...

- **Informações técnicas sobre o inode em sistemas Ext**
  - A estrutura ocupa 128 bytes (**160 bytes em sistemas mais recentes**)
  - Possui **15 ponteiros** para endereços de blocos: **12 primeiros blocos de dados do arquivo + 3 blocos indiretos**
  - Blocos de tamanho **4 KB**

Inodes

- **Vantagens**
  - Permite lidar com arquivos muito **pequenos** e muito **grandes** de forma **eficiente**
  - **Acesso rápido a arquivos pequenos**, pois seus endereços de blocos são armazenados diretamente no **inode**

- **FAT vs inode table**
  - **FAT**: tamanho **proporcional** à **quantidade de blocos do disco** (FAT terá n entradas em um disco com n blocos)
  - **Inode table**: tamanho **proporcional à quantidade de arquivos que podem estar abertos simultaneamente** (não depende do tamanho do disco)

3 Implementação de diretórios

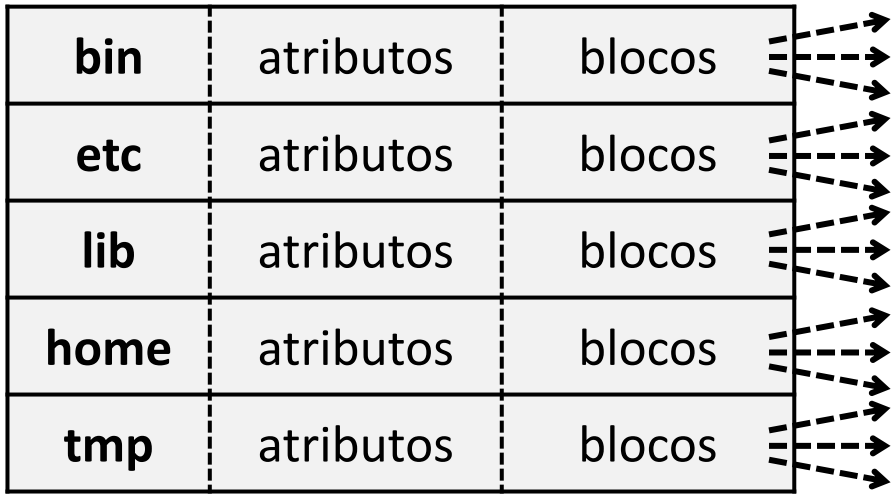
Implementação de diretórios

- A função do sistema de diretórios é **mapear um nome de arquivo (ou caminho)** em uma **informação que permite localizar os seus dados**
- Essa informação depende do **método de alocação de blocos no disco**
  - **Alocação contígua**: endereço de disco onde começa o arquivo
  - **Lista encadeada**: número do primeiro bloco
  - **Inodes**: número do inode

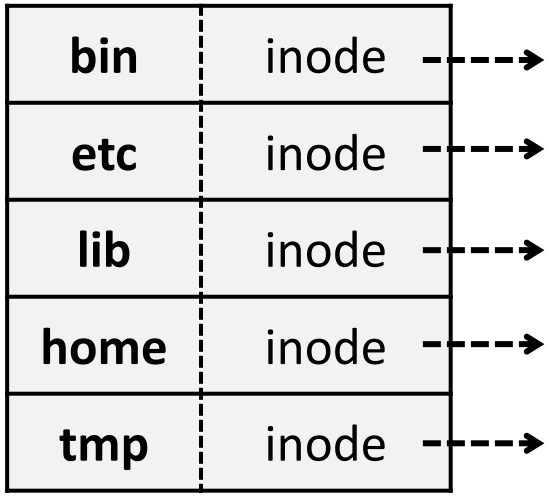
Implementação de diretórios

Duas formas de armazenar atributos de arquivos

Nas entradas de diretório



Em inodes



4 Gerenciamento de blocos livres

Gerenciamento de blocos livres

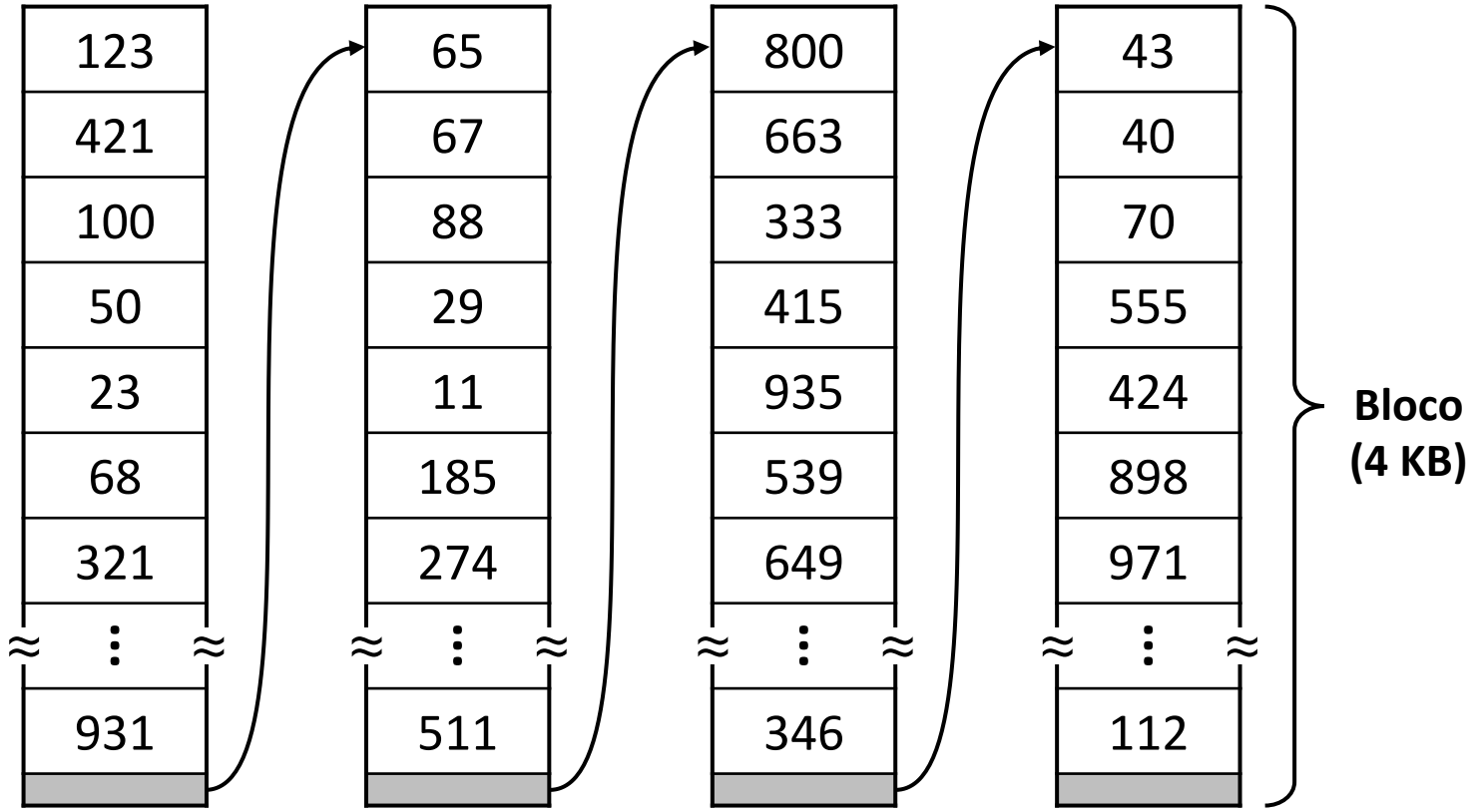
- O sistema de arquivos necessita manter o **controle de blocos livres em disco**
- **A alocação de blocos acontece quando:**
  - Um novo arquivo é **criado**
  - O tamanho do arquivo **ultrapassa o tamanho do bloco**
- **Duas abordagens clássicas**
  - Lista encadeada de blocos livres
  - Mapa de bits (bitmap)

Lista encadeada de blocos livres

- Armazena **números de blocos livres** em blocos do disco
- Esses blocos são **encadeados**, formando uma **lista encadeada**

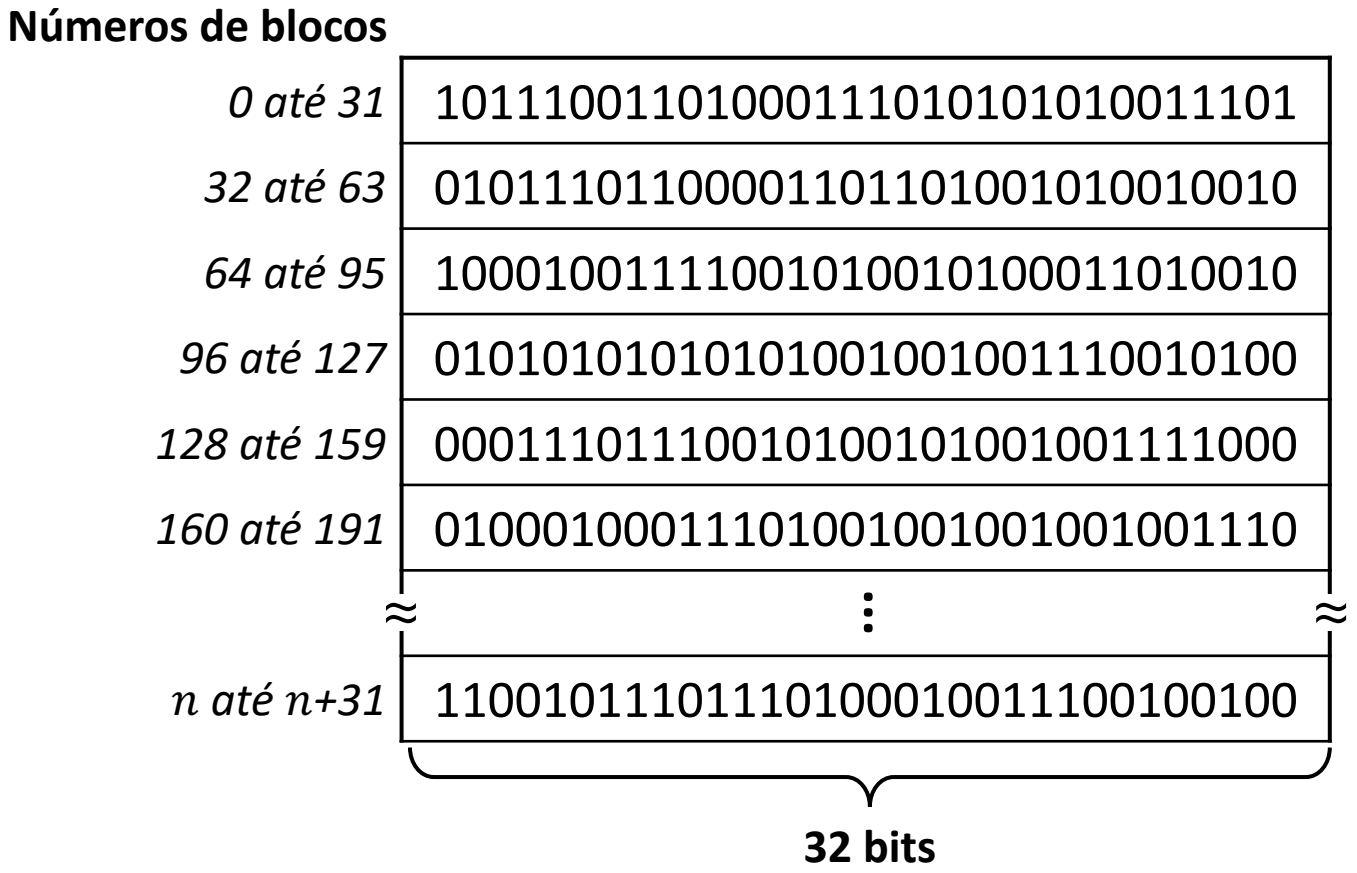
Exemplo

- Blocos de 4 KB
- Número de bloco de 32 bits
- Cada bloco conterá 1023 números de blocos livres e um ponteiro para bloco seguinte



Mapa de bits (bitmap)

- Um disco com n **blocos** requer um mapa de n **bits**
- Cada bit indica se o bloco está **livre (0)** ou **ocupado (1)**



Adaptado de *TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. Rio de Janeiro: Pearson Prentice Hall, 2010. xiii, 653 p. ISBN 9788576052371.*

Obrigado pela atenção!

Dúvidas? Entre em contato:

- marcio.castro@ufsc.br
- www.marciocastro.com

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LaPeSD

Distributed Systems Research Lab

www.lapesd.inf.ufsc.br