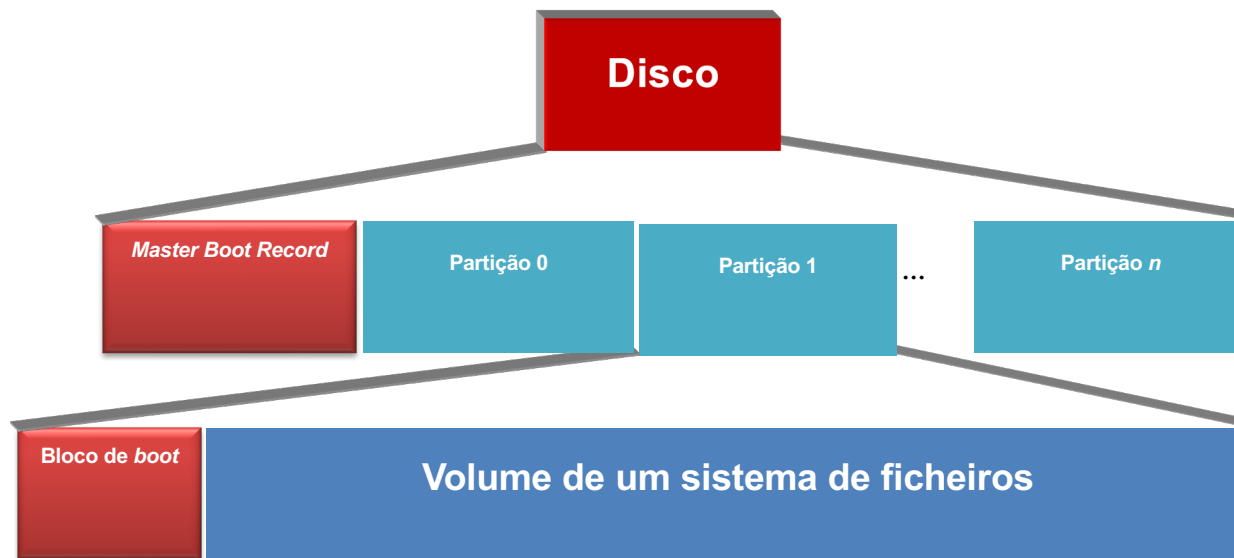




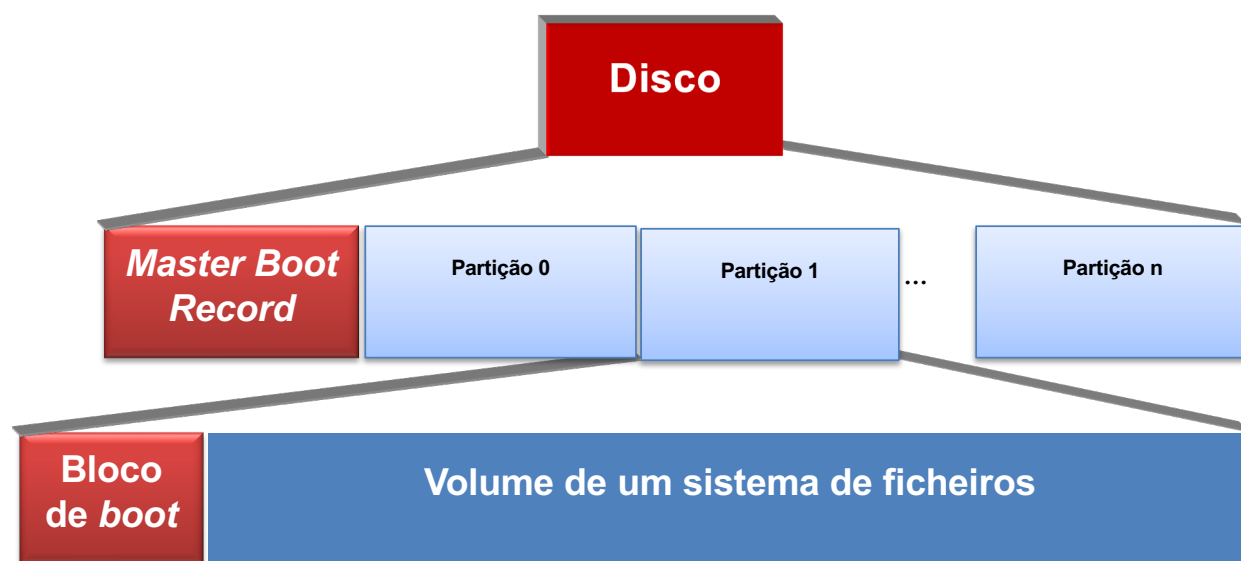
# Organização dos Ficheiros no Disco

# Visão de um Dispositivo do Tipo Disco



- Uma **partição** é uma subdivisão lógica de um dispositivo físico (disco, pen, etc.), dividida em blocos de tamanho fixo.
- Uma partição é vista pelo sistema operativo como um vetor de blocos ordenado a partir do número zero
- Cada partição é um contexto independente das outras, não existindo ficheiros repartidos por partições diferentes

# Visão de um Dispositivo do Tipo Disco



- O *Master Boot Record* (MBR) e o bloco de *boot* são entidades fundamentais para localizar e executar um sistema operativo
- O MBR possui código, geralmente independente do sistema operativo, que localiza a partição que contém o sistema operativo a executar e transfere a execução para o código existente no primeiro bloco dessa partição – o bloco de *boot*.



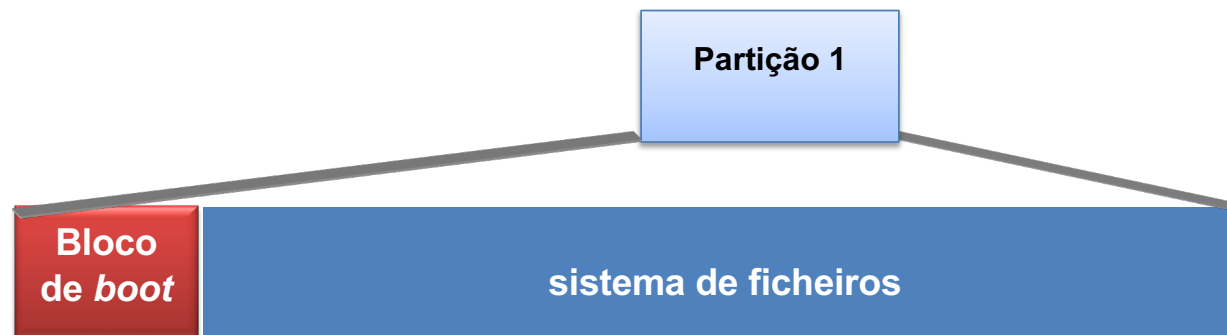
# *Boot*

- O bloco de ***boot*** possui um programa específico de cada sistema operativo que sabe ler o sistema de ficheiros onde o sistema operativo se encontra, carregar o sistema operativo e executá-lo.
- Quando só existe uma partição com sistema operativo, o código do MBR escolhe imediatamente essa partição para transferir a execução durante a fase de *boot*
- Quando existem mais partições, usualmente é o utilizador que escolhe a partição a executar. Neste caso o código existente no MBR diz-se que é um *boot loader* porque permite escolher entre vários sectores de *boot*, de várias partições, para executar.



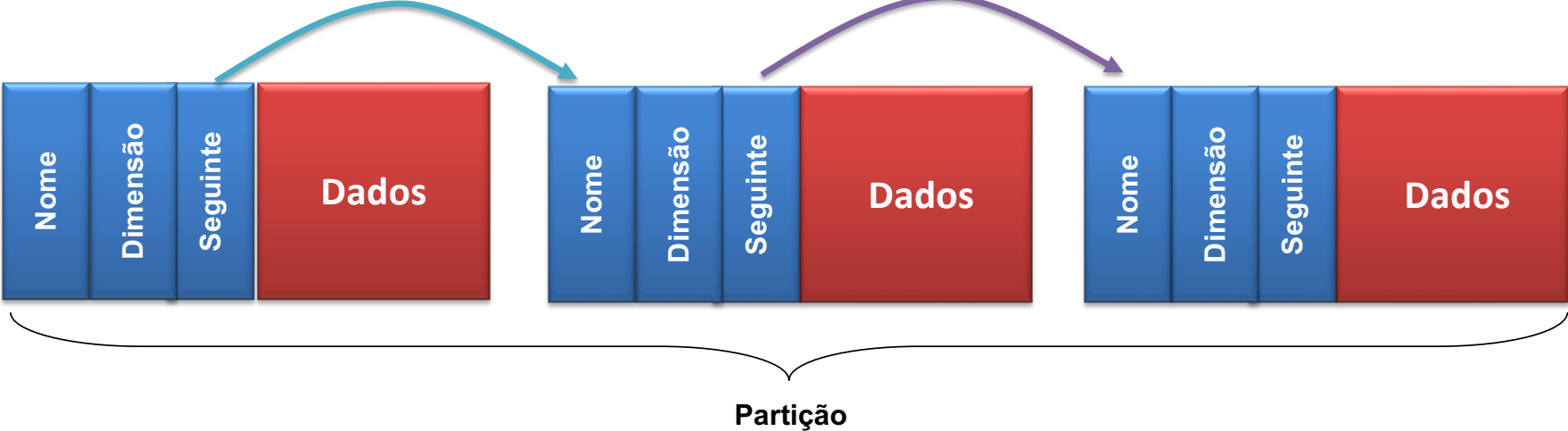
# Organizações dos Ficheiros

- Dada esta organização do disco o problema interessante do ponto de vista de sistema operativo é como organizar os ficheiros
- Vamos ver diferentes alternativas que correspondem a evoluções de sistemas reais



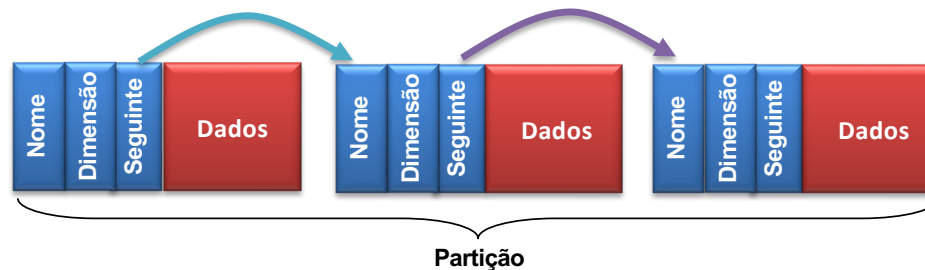


# Alternativa 1: Organização em Lista



# Organização em Lista

- Forma mais simples de organizar um sistema de ficheiros
- Cada ficheiro é constituído por um registo de dimensão variável com quatro campos



- No caso dos sistemas de ficheiros em CD/DVD, que só podem ser escritos uma vez, todos os ficheiros ficam compactados uns a seguir aos outros
  - No caso de sistemas onde é possível apagar ficheiros, é necessário manter também uma lista de espaços livres

**Desvantagens ?**



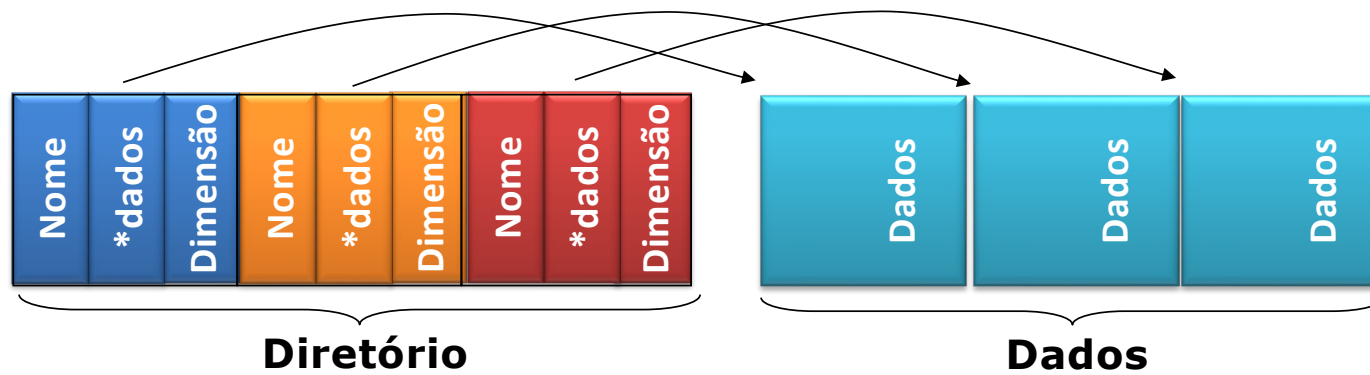
## Alternativa 1: desvantagens

- Tempo necessário para localizar um ficheiro através do seu nome
- Ficheiros que mudam de tamanho ou são apagados são problemáticos:
  - Espaço ocupado por cada ficheiro é contínuo
  - Fragmentação da memória

Todos estes problemas aplicam-se a sistemas de ficheiros em DVD?

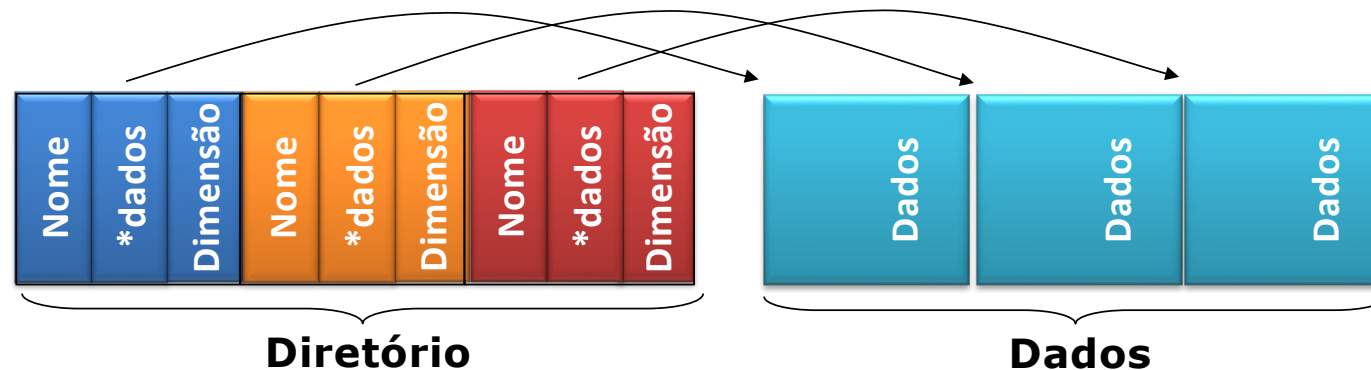


## Alternativa 2



## Alternativa 2

- Solução para a primeira desvantagem:
  - criando um directório único onde todos os nomes dos ficheiros estão juntos



- os nomes dos ficheiros ficam perto uns dos outros no disco
- aumenta a eficiência da procura de um ficheiro dado o seu nome

# Organização em Lista – desvantagens (cont.)

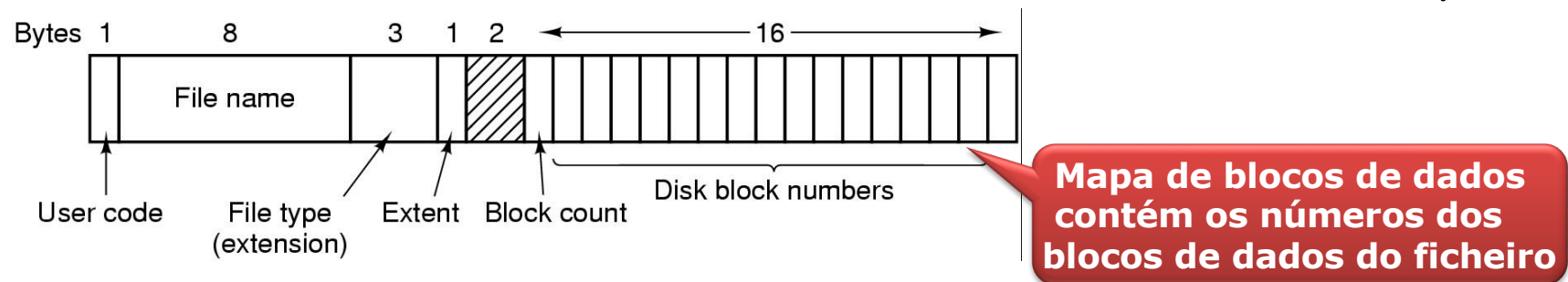
- Solução para a segunda desvantagem: dividir os dados de cada ficheiro em blocos de dimensão fixa
- O sistema de ficheiros do CP/M (um dos primeiros para PCs, 1977) utilizava uma estrutura deste tipo



CP/M advertisement in the November 29, 1982 issue of InfoWorld magazine

# Sistema de Ficheiros do CP/M

- Estrutura de uma entrada do diretório do sistema de ficheiros do CP/M:



- Neste sistema:
  - cada bloco possuía 1 Kbyte (por omissão)
  - mapa de blocos com 16 entradas, dimensão máxima de um ficheiro = 16 KBytes
- Como aumentar a dimensão máxima dos ficheiros?
  - Aumentar o mapa de blocos → ineficiente para fich. pequenos
  - Aumentar o tamanho dos blocos → maior fragmentação interna



# Organização do Disco em Blocos

- A gestão do disco é feita em blocos de tamanho fixo com o objetivo de otimizar o acesso ao disco.
- Um bloco é a unidade mínima que pode ser indexada diretamente pelo sistema operativo
- Tamanho dos blocos
  - Quanto maior for o bloco maior é a taxa de transferência bruta pois os tempos de posicionamento e latência de sectores consecutivos são quase nulos, mas a taxa de transferência efetiva depende do número de *bytes* dentro do bloco com informação útil.
  - Se um bloco estiver apenas meio cheio a sua transferência efetiva é metade da transferência bruta, o que significa que quanto maiores forem os blocos, maiores serão as diferenças entre as taxas de transferência bruta e efetiva

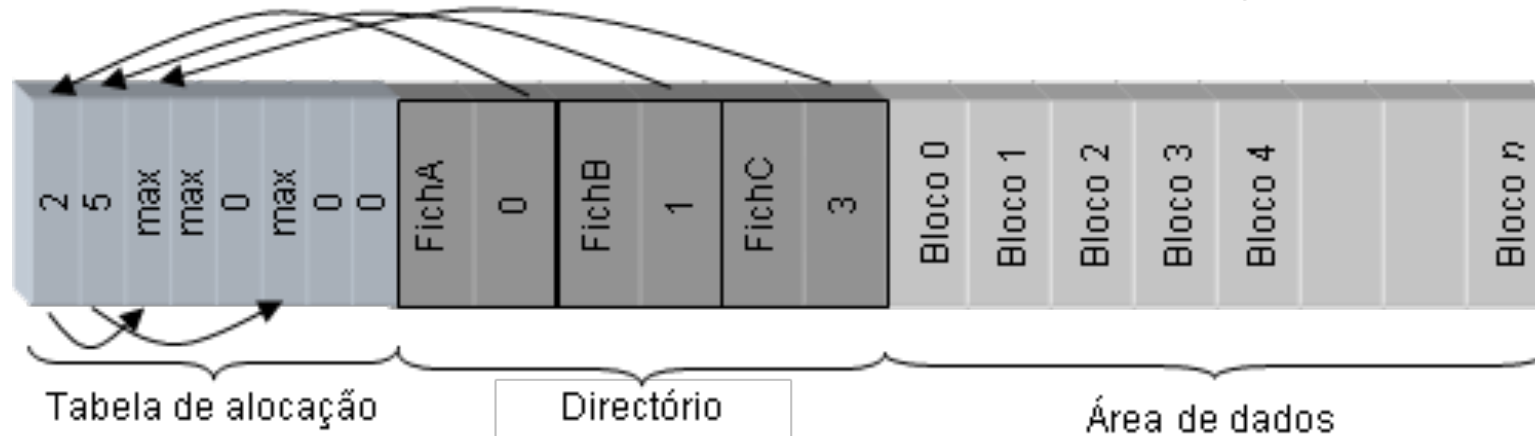
# Sistema de Ficheiros do MS-DOS

- Evoluiu a partir do sistema CP/M
- Possui uma estrutura de sistema de ficheiros semelhante
- Em vez de um mapa de blocos por ficheiro, no MS-DOS existe:
  - uma tabela de blocos global partilhada por todos os ficheiros
  - esta tabela única é tão representativa da estrutura do sistema de ficheiros que deu origem ao nome do sistema de ficheiros mais popular que a usa: **o sistema de ficheiros FAT** (Tabela de Alocação de Ficheiros — *File Allocation Table*).



The original MS-DOS advertisement in 1981

# Sistemas de Ficheiros do Tipo FAT



- A partição contém três secções distintas:
  - a **tabela de alocação (File Allocation Table, FAT)**,
  - uma diretoria com os nomes dos ficheiros presentes no sistema de ficheiros, e
  - uma secção com o espaço restante dividido em blocos, de igual dimensão, para conter os dados dos ficheiros

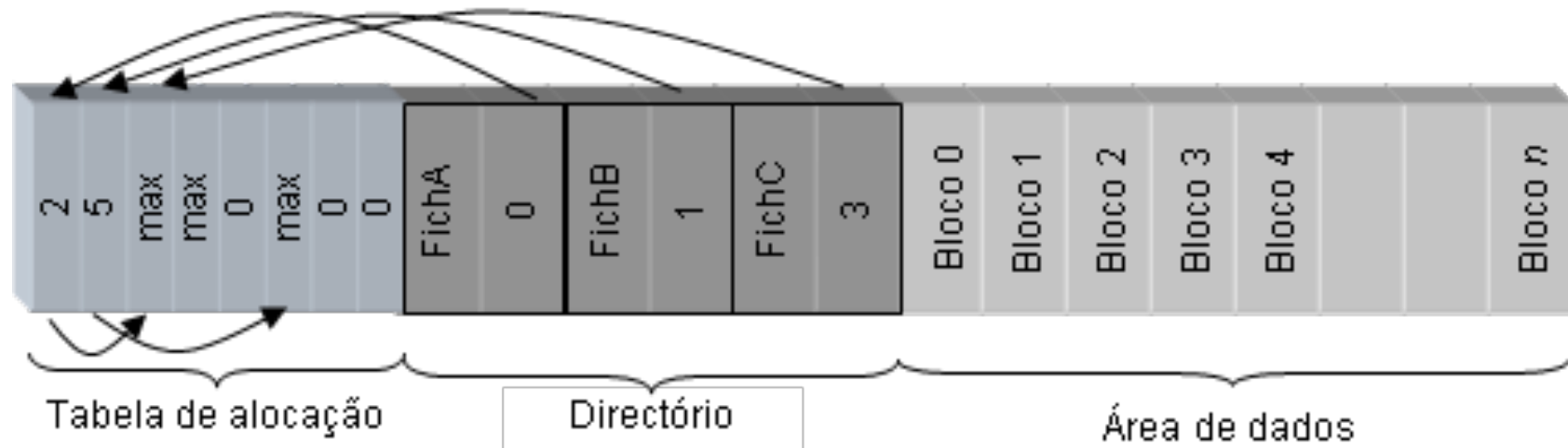


# Sistemas de Ficheiros do Tipo FAT (cont.)

- FAT é vetor composto por  $2^n$  inteiros de  $n$  bits
  - Designado de FAT-16 ( $n=16$ ) ou FAT-32 ( $n=32$ ), etc.
- Dimensionado para:
  - Caber em memória RAM (FAT carregada do disco em RAM quando o FS é montado)
  - Ter tantas entradas quanto o número de blocos de dados na partição em disco
- As entradas da FAT:
  - com o valor zero indicam que o respectivo bloco está livre,
  - com valores diferentes de zero indicam que o respectivo bloco faz parte de um ficheiro



# Sistemas de Ficheiros do Tipo FAT (cont.)



- Os blocos de um ficheiro são determinados assim:
  - 1º bloco: indicado por um número na respectiva entrada no directório.
  - restantes blocos: referenciados em **lista ligada** pelas entradas da FAT



# Desvantagens do FAT

- Elevada dimensão da FAT quando os discos têm dimensões muito grandes:
  - Por exemplo, uma partição 1 *Tbyte*
  - Usando FAT-32 e blocos de 4 *Kbytes*...
  - ...a FAT pode ocupar 1 *GByte* ( $1\text{TBytes}/4\text{KBytes} \times 4 \text{ bytes}$ )
- Tabelas desta dimensão não são possíveis de manter em RAM permanentemente:
  - Ler à FAT do disco, prejudica muito o acesso à cadeia de blocos de um ficheiro



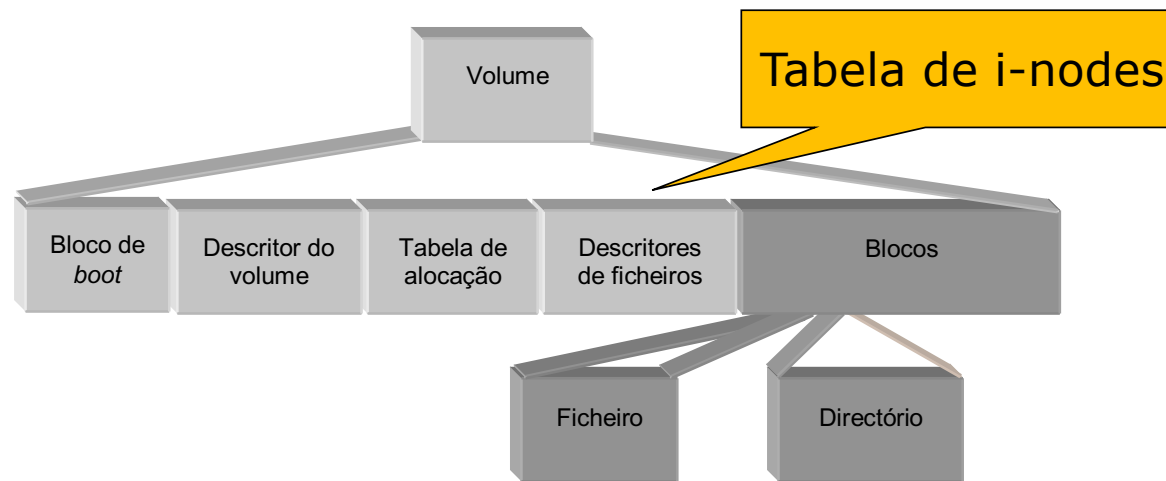
## Alternativa 5: Organização com Descritores Individuais de Ficheiros (i-nodes)

- Manter a descrição do ficheiro num descritor próprio de cada ficheiro, chamado i-node
  - Exemplos de atributos incluídos no i-node: tipo de ficheiro, dono, datas de últimos acessos, permissões, dimensão, localizações dos blocos de dados
  - É a estrutura que está entre as entradas dos diretórios que referenciam o ficheiro e os seus blocos de dados
- Vantagem: podem existir várias entradas de diretório a apontar para o mesmo ficheiro
  - Noção de *hard link*



# Organização com Descritores Individuais de Ficheiros (i-nodes)

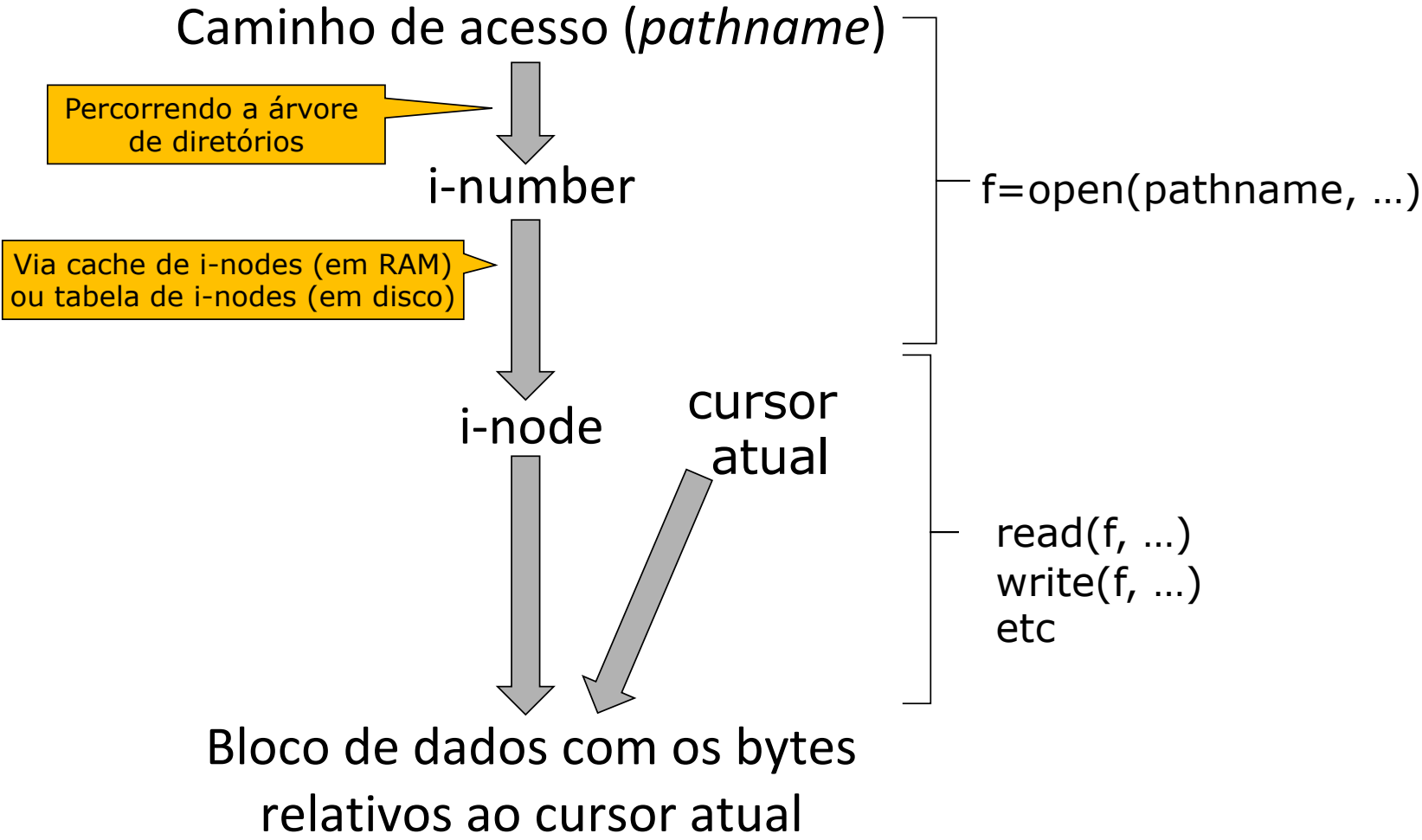
- Os i-nodes são guardados numa estrutura especial de tamanho fixo antes dos blocos de dados



- No Linux tem o nome de tabela de *inodes*
- No Windows tem o nome:
  - MFT (*Master File Table*).
- O número máximo de ficheiros numa partição é dado pelo número máximo de i-nodes nessa tabela



# A sequência de passos para aceder ao conteúdo de um ficheiro





# Organização com Descritores Individuais de Ficheiros (i-nodes)

- Um ficheiro é univocamente identificado, dentro de cada partição, pelo número de i-node (muitas vezes chamado i-number)
- Os directórios só têm que efetuar a ligação entre um nome do ficheiro e o número do seu descritor

	NÚMERO DO INODE			DIMENSÃO DO REGISTO		DIMENSÃO DO NOME	TIPO	NOME							
0		54		1	2	1	2	.	\0	\0	\0				
12		79		1	2	2	2	.	.	\0	\0				
24		23		1	6	6	1	c	a	r	l	o	s	\0	\0
40		256		1	6	7	1	m	a	r	q	u	e	s	\0



## Percorrer a árvore de diretórios

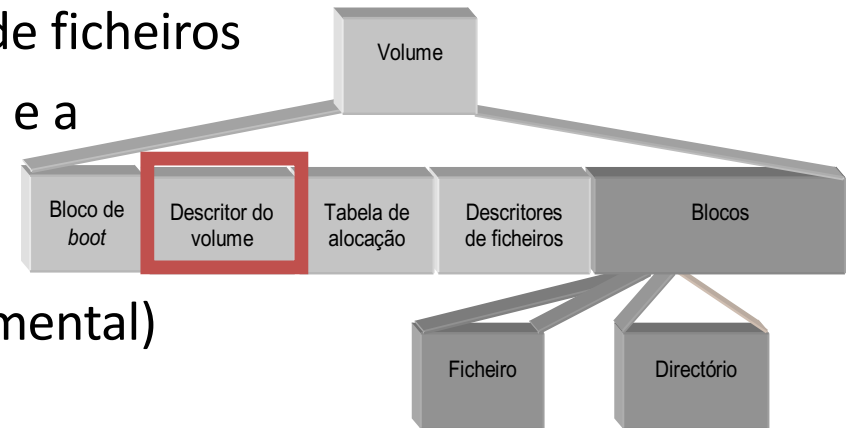
Recursivamente para  
cada elemento do *pathname*

1. Começar pelo diretório raíz
  - i-number tem valor pré-conhecido (e.g., i-num=2)
2. Dado o i-number, obter o i-node do diretório
  - Na cache de i-nodes (em RAM) ou na tabela de i-nodes (em disco)
3. A partir do i-node, descobrir os índices dos blocos de dados com o conteúdo do diretório
4. Ler cada bloco do diretório e pesquisar nele uma entrada com o próximo nome do *pathname*
5. Assim que seja encontrada, a entrada indica o i-num do próximo nome
6. Repetir a partir do passo 2 para este novo nome

# Descritor do volume

- Descritor do Volume:

- possui a informação geral de descrição do sistema de ficheiros
- por exemplo, a localização da tabela de descritores e a estrutura da tabela de blocos livres
- é geralmente replicado noutros blocos (a informação nele guardada é de importância fundamental)
- se se corromper pode ser impossível recuperar a informação do sistema de ficheiros



- Implementação do descritor de volume:

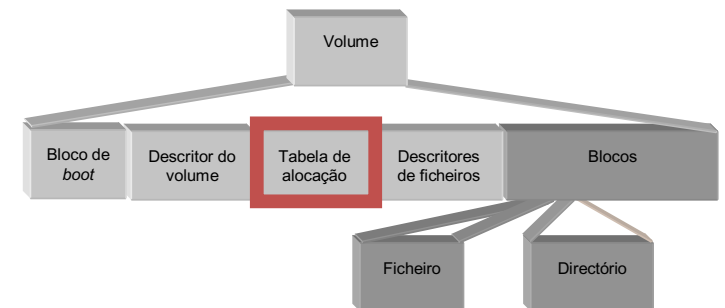
- Unix - bloco especial denominado superbloco
- NTFS - ficheiro especial
- FAT – a informação em causa é descrita directamente no setor de boot





# Tabela de Blocos Livres (ou Tabela de Alocação)

- Mantém um conjunto de estruturas necessárias à localização de blocos livres:
  - i.e. blocos da partição que não estão ocupados por nenhum bloco de nenhum ficheiro.
- Pode ser um simples **bitmap**:
  - um bit por cada bloco na partição,
  - indica se o respetivo bloco está livre ou ocupado
- Tabela de blocos livres desacoplada dos i-nodes tem vantagens:
  - é possível ter estruturas muito mais densas (a tabela de blocos livres possui, usualmente, apenas um bit por cada bloco)
  - pode-se organizar a tabela de blocos livres em várias tabelas de menor dimensão para blocos adjacentes





# Sistema de ficheiros Ext

Principal Sistema de ficheiros do Linux  
Sistema referência para outros sistemas de  
ficheiros atuais



## i-node (*index node*)

- Meta-dados do ficheiro
- Localização dos dados do ficheiro
  - Índices do 1º bloco, do 2º bloco, etc.

Campo	Descrição
i_mode	Tipo de ficheiro e direitos de acesso
i_uid	Identificador do utilizador
i_size	Dimensão do ficheiro
i_atime	Tempo do último acesso
i_ctime	Tempo da última alteração do inode
i_mtime	Tempo da última alteração do ficheiro
i_dtime	Tempo da remoção do ficheiro
i_gid	Identificador do grupo do utilizador
i_links_count	Contador de hard links
i_blocks	Número de blocos ocupado pelo ficheiro
i_flags	Flags várias do ficheiro
i_block[15]	Vector de 15 unidades para blocos de dados
	Outros campos ainda não utilizados

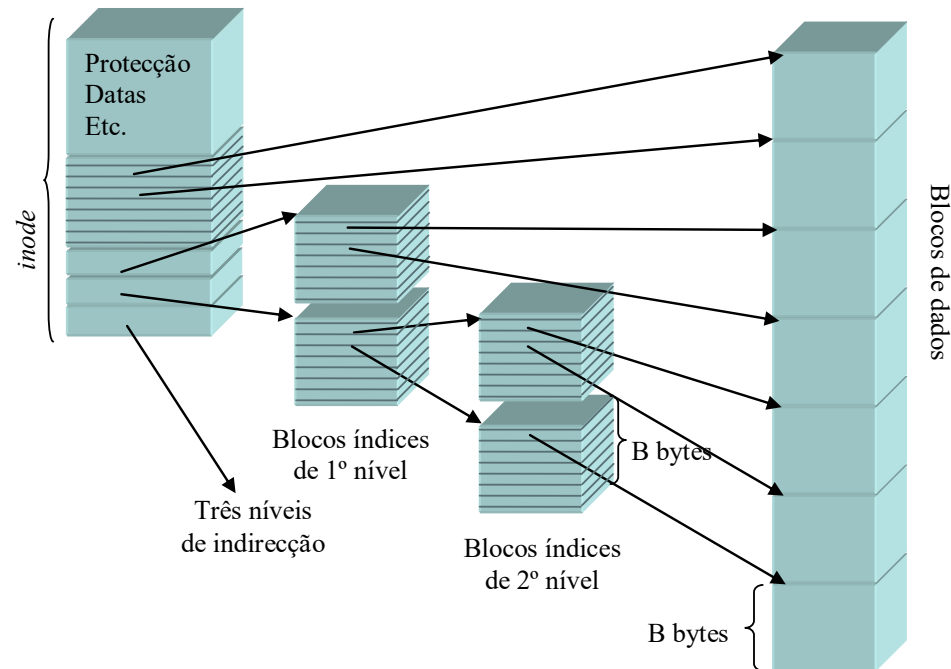


# Exemplo de FS que usa i-nodes:

## ext3

- Índice dos blocos do ficheiro é mantido num vetor *i\_block* do i-node, com 15 posições
  - 12 entradas diretas
  - Caso dimensão > 12 blocos, 3 últimas posições do vector contêm referências para blocos com índices para outros blocos
    - Primeira entrada é indireção de 1 nível
    - Segunda é indireção de 2 níveis
    - Terceira é indireção de 3 níveis
- Só se usam as entradas (e blocos de índices) necessários

# Exemplo de FS que usa i-nodes: ext3



dimensão máxima de um ficheiro =  $B \times (12 + B/R + (B/R)^2 + (B/R)^3)$

B é a dimensão em bytes de um bloco de dados

R é a dimensão em bytes de uma referência para um bloco

Com blocos de 1 Kbyte e referências de 4 byte, a dimensão máxima de um ficheiro é ~16 Gbyte



# Tabelas de Inodes e Bitmaps

- Cada partição tem um número máximo de *i-nodes*:
  - que servem para armazenar os *i-nodes* de todos os ficheiros da partição.
  - logo, existe um número máximo de ficheiros, correspondente à dimensão máxima da tabela de *i-nodes*
- Dentro de uma partição:
  - um *i-node* é univocamente identificado pelo índice dentro da tabela de *inodes*.
- Para além da tabela de *inodes* existem em cada partição ainda duas outras tabelas:
  - o *bitmap* de *i-nodes* – posições dos *i-nodes* livres
  - o *bitmap* de blocos, – posições dos blocos livres

# Conclusões

- A organização do disco é em blocos de tamanho fixo.
- Diversas zonas do discos são atribuídas a funções específicas, bloco de *boot*, partições, diretórios, blocos de dados
- A indexação dos blocos de dados de um ficheiro é uma importante parte da organização porque tem elevado impacto no desempenho e na fragmentação do disco
- No Unix a estrutura de indexação é mantido nos *inodes* procurando um equilíbrio entre acesso rápido a ficheiros pequenos e possibilidade de ter ficheiro muito grandes