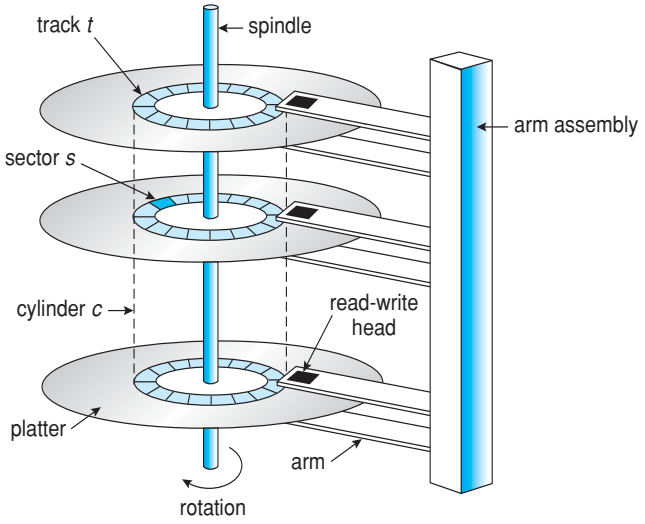
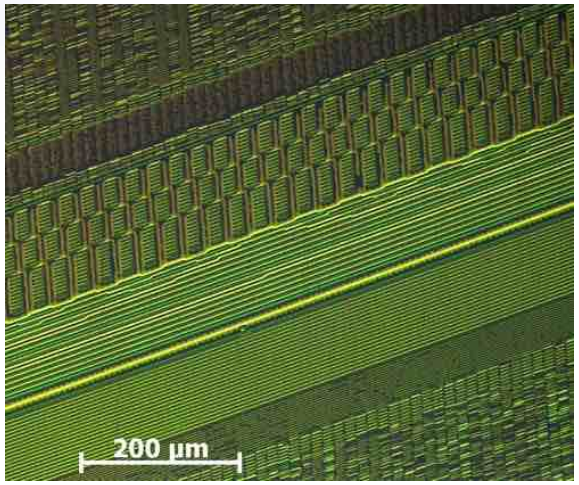
Sistema de Ficheiros

Introdução

- **Problema**: como armazenar muita informação de forma permanente num suporte que o permita? (disco, CD, usbpen, ...)
- **Solução**: sobrepor à organização física do "meio" (sectores, ...) uma organização em "peças" de informação lógica: ficheiros
- É da responsabilidade do SO criar esta organização lógica
- Ficheiro
  - Conjunto de dados persistentes, geralmente relacionados, identificados por um nome
  - Organizado em hierarquia de pastas

A realidade


track t
spindle
arm assembly
sector s
cylinder c
read-write head
platter
arm
rotation

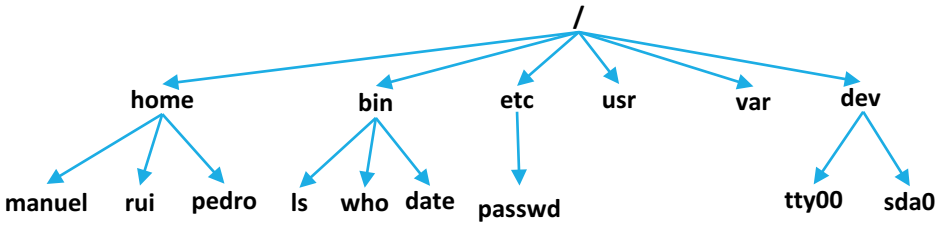

200 µm

Sistema de Ficheiros (*File System*)

- Forma como o SO organiza o suporte físico em ficheiros
  - formatação do disco, localização dos ficheiros no disco, ...
- Composto por um conjunto de entidades
  - Sistema para organização de nomes para identificação dos ficheiros
  - Uma interface programática para comunicação com os processos
- Proporciona os mecanismos para os "utilizadores" (programas) lidarem com esses ficheiros
  - acesso: criação, leitura/escrita
  - proteção
  - *buffering* ...

Organização dos ficheiros

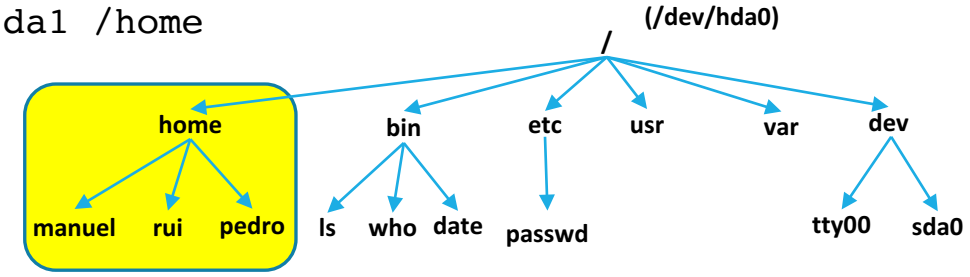
- Formas de organizar os ficheiros
  - Apenas um diretório global e atribuir um nome a cada ficheiro
    - usado nos primeiros sistemas de processamento por lotes
    - problema de colisão de nomes
    - uma primeira evolução: atribuir um diretório diferente para cada utilizador
  - Organização hierárquica, na forma de uma árvore
    - Proposta no sistema MULTICS pela primeira vez
    - Solução mais comum nos sistemas atuais



Hierarquia de nomes

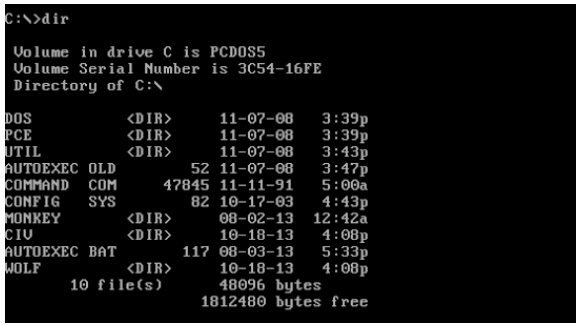
- Quer o Windows quer o Unix têm um espaço de nomes uniforme
  - ficheiros, diretórios, dispositivos são referenciados usando a mesma sintaxe
- Em ambos os casos existe uma única raiz de nomes
  - todos os nomes de ficheiros, diretórios e dispositivos começam por '/' (Unix) ou '\' (Windows)
- *Mount*: é uma operação consiste em ligar a raiz do novo sistema de ficheiro a um diretório do sistema de ficheiros raiz.

```
mount –t <filesystem> /dev/sda1 /home
```



Ficheiros - None

- Extremamente importante para o utilizador
  - Um dos aspetos do sistema de ficheiros visível do exterior
  - O primeiro aspeto visível de utilização de um ficheiro
    - `cp um_ficheiro outro_ficheiro`
  - Dimensão
    - MS/DOS – tamanho fixo – 8+3 (extensão)
    - Linux – tamanho variável (limite de 255 caracteres)
  - Extensão
    - Formal ou informalmente indicia a natureza (ou conteúdo) do ficheiro
  - Case sensitive/insensitive – distinguir letras maiúsculas e minúsculas
    - Unix: case sensitive
    - Windows: case insensitive (por razões de compatibilidade)



Ficheiros – Extensão

- Unix
  - As extensões são convencionais (tratadas pelo utilizador) e, eventualmente, forçadas pelos programas que tratam determinado tipo de ficheiros
    - exemplos: compilador .c .o; .java
- Windows
  - Extensões têm significado formal no SO
    - ex: ficheiro.exe – um executável
  - Pode-se relacionar uma extensão com uma aplicação (registry)
    - Permite invocar uma aplicação (isto é executar um programa) abrindo um ficheiro com a extensão correspondente
      - ex: ficheiro.doc => executar programa word.exe (passando o ficheiro como argumento)
- Exemplos: .c .cpp .h .java, .htm .html .xml, .bmp .jpeg .mp3 .mpg, .Z, .gz, .ZIP

Ficheiros – Conteúdo

- Ficheiros de texto
  - Conteúdo físico
    - Sequência de caracteres ASCII, incluindo alguns caracteres especiais: 10 (Line feeder), 13 (Carriage return)
    - Visível diretamente, exemplo:
      - `cat /etc/passwd`
  - O conteúdo funcional pode ser o mais diverso
    - exemplo: código em c; script shell; dados xml
- Ficheiros binários
  - Conteúdo físico não interpretável como um conjunto de caracteres ASCII
  - Não é visível diretamente
    - exemplo: `cat /bin/cat; reset`
  - Tratáveis apenas por programas (ou os próprios ficheiros são programas)

Ficheiros – Conteúdo

- Exemplo: registo de dados (número 37)

Formato de texto

```
#include <stdio.h>
int main(){
    int i = 37;
    FILE *fp = fopen("teste.txt", "w");
    fprintf(fp, "%d", i );
    fclose(fp);
}
```

```
➢  ls -la teste.txt
-rw-r--r--  1 fmmb  staff  2  2 Out 19:13 teste.txt
➢  hexdump teste.txt
0000000 33 37
```

Resultado:
ficheiro com 2 bytes
código do caractere '3' e
código do caractere '7'

Formato binário

```
#include <stdio.h>
int main(){
    int i = 37;
    FILE *fp = fopen("teste.dat", "w");
    fwrite(&i, sizeof(i), 1 , fp );
    fclose(fp);
}
```

```
➢  ls -la teste.dat
-rw-r--r--  1 fmmb  staff  4  2 Out 19:17 teste.dat
➢  hexdump teste.dat
0000000 25 00 00 00
```

Resultado:
ficheiro com 4 bytes
representando o número
37 em binário
(numa máquina com inteiros de 32 bits)

Ficheiros – Acesso

- Ficheiros de acesso sequencial
  - Localizar lendo o ficheiro, desde o início até à posição pretendida
  - Historicamente ligado às bandas magnéticas
  - Ex: é dado um ficheiro de texto com vários números inteiros, um em cada linha – para obter o 13º número inteiro:

```
#include <stdio.h>
int main() {
    int i, n;
    FILE *fp = fopen("teste.txt", "r");
    for (i=1; i<= 13; i++)
       fscanf(fp, "%d", &n);
}
```

Ficheiros – Acesso

- Ficheiros de acesso direto
  - Possibilidade de localizar uma posição ("seek"), com base numa chave ou outra indicação
  - Ex: é dado um ficheiro de texto com vários números inteiros – obter o 13º número:

```
#include <stdio.h>
int main() {
    int n;
    FILE *fp = fopen("teste.txt", "rb" );
    fseek(fp, 12 * sizeof(int), SEEK_SET );
    fread(&n, sizeof(int), 1, fp);
}
```

- Ficheiros de acesso por chave (usado em BDs)

Ficheiros – Atributos

- Atributos
  - Características do ficheiro (além do nome), para efeitos de controlo e administração
  - Podem variar de sistema de ficheiros para sistema de ficheiros
- Alguns atributos significativos (maior utilidade)
  - Dono (Owner) – utilizador dono / criador do ficheiro
    - geralmente corresponde a quem o criou. controlo de acessos, quotas, etc.
  - Proteção – permissões de acesso ao ficheiro
    - quem pode aceder ao ficheiro e quais as operações que pode realizar
  - Datas – data de criação, última leitura, última escrita
    - gestão, procura por "tempo"; suporte do make
  - Dimensão – dimensão atual do ficheiro
    - gestão / quantificação do espaço livre/ocupado
  - Flags extra – *hidden*, *system*, etc.
    - gestão do acesso

Ficheiros – Operações

- Operações de manipulação de um ficheiro
  - read – ler o conteúdo de um ficheiro
    - `cat teste ; fgets(...);`
  - write – escrever (alterar) o conteúdo de um ficheiro
    - `cat > teste ; fprintf (...);`
  - create – criação do ficheiro (vazio)
    - `cat > novo_ficheiro ; fopen(..., "w" );`
  - open – abrir um ficheiro para determinada operação
    - `fopen (... , "r");`
  - close – fechar um ficheiro depois de concluído um conjunto de operações
    - `fclose(...);`

Ficheiros – Operações

- Operações de manipulação de um ficheiro
  - append – acrescentar ao conteúdo anterior
    - `cat >> teste ; fopen(..., "w+");`
  - seek – posicionamento para acesso directo
    - `fseek (...)`
  - get/set attributes – obter / alterar atributos
    - `chmod ; chmod (...); stat (...);`
  - rename – alterar o nome de um ficheiro
    - `mv .... ; rename(...);`
  - delete – remover um ficheiro
    - `rm antigo.txt; remove(...);`

Ficheiros – Operações

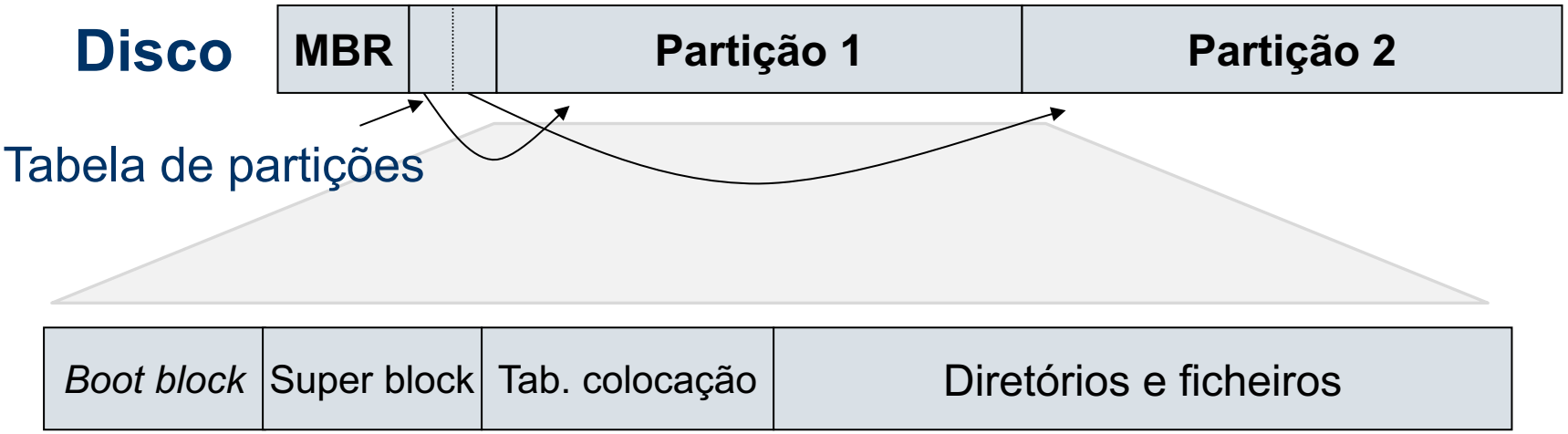
- Lock
  - bloqueio de todo / parte do ficheiro
  - mecanismo de exclusão para acesso simultâneo
    - lock READ (não exclusivo)
      - pode ser feito por vários processos
      - impede o lock WRITE por um outro processo
    - lock WRITE (exclusivo)
      - impede qualquer outro lock (read ou write) por um outro processo
- *memory mapped files*
  - mapeamento do ficheiro no espaço de endereçamento do processo;
  - permite o acesso ao ficheiro através de "variáveis"

Diretórios

- Árvore de diretórios e ficheiros
  - Diretórios (nós) – contêm uma lista de outros diretórios ou ficheiros (folhas)
    - Nome do ficheiro – nome único na árvore de ficheiros – caminho desde o diretório principal (raiz) até diretório onde o ficheiro se encontra
    - Ou nome relativo – caminho a partir de um diretório intermédio ("corrente")
  - Link – representação do mesmo ficheiro com mais do que um nome, em mais que um diretório
    - *soft* – referência simbólica para um ficheiro real definido noutro diretório
    - *hard* – representação do mesmo ficheiro em mais que um diretório

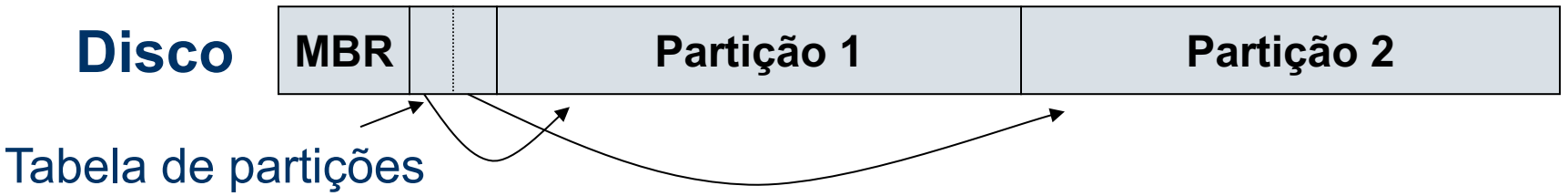
Organização lógica

- Organização típica do disco
  - A formatação de baixo nível divide o disco em sectores (blocos)
  - Em cima dessa formatação, o disco é dividido em partições (uma ou mais por disco)
  - Cada partição pode ter um sistema de ficheiros diferente



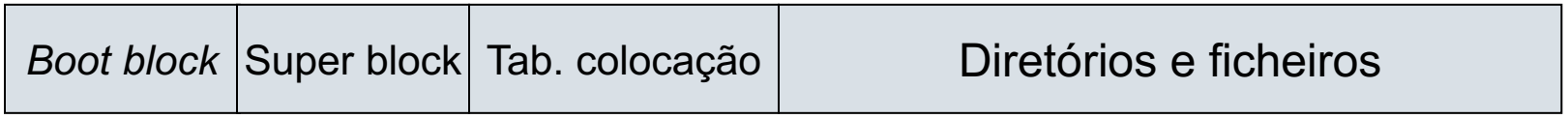
Organização lógica

- Código que efetua o *boot* do sistema (parte do BIOS)
  - Assume que a informação sobre as partições existentes estejam no início do disco, no *MBR*
- *MBR - Master Boot Record*
  - Possui código, geralmente independente do sistema operativo, que localiza a partição que tem o sistema operativo e transfere a execução para o primeiro bloco dessa partição (*boot block*)
- Tabela de partições – lista das partições existentes no disco



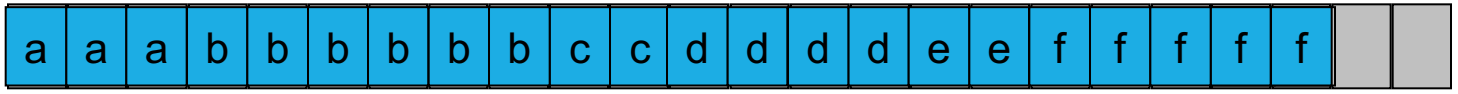
Organização lógica

- Partição: organização típica
  - Boot block
    - Programa de arranque do sistema operativo que sabe ler o sistema de ficheiros onde o SO se encontra instalado, carrega o SO e executa-o
  - Super-block
    - Informação característica do sistema de ficheiros instalado
  - Tabelas / estruturas de colocação
    - Informação relativa aos blocos do disco livres/ocupados com ficheiros
    - Elementos de controlo e gestão próprios de cada sistema de ficheiros
      - Unix: i-nodes; MS-DOS: FAT
  - Diretórios e ficheiros
    - Sectores ocupados com os ficheiros e diretórios armazenados no disco



Implementação do Sistema de Ficheiros

- Colocação contígua
  - Armazenamento de cada ficheiro num conjunto de blocos contíguos
  - Vantagens:
    - Implementação simples – para cada ficheiro, basta o SO saber o bloco inicial e final (ou nº de blocos)
    - Eficiente em termos de leitura – sempre bloco contíguos



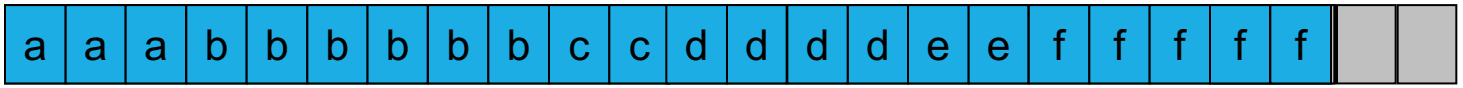
Partição

Tabela de colocação

Ficheiro	Início	Nº de blocos
a	0	3
b	3	6
c	9	2
d	11	4
e	15	2
f	17	5

Implementação do Sistema de Ficheiros

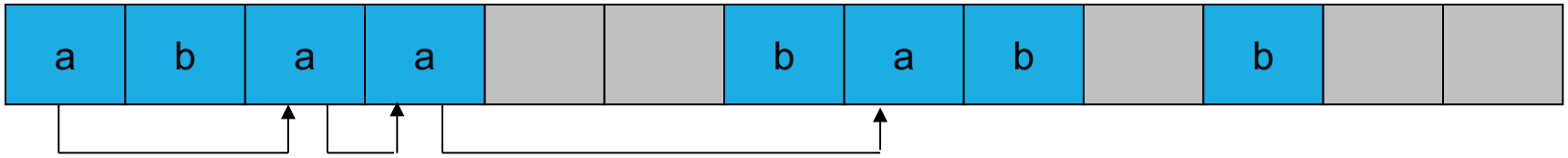
- Colocação contígua
  - Desvantagem
    - Fragmentação resultante da remoção / criação de novos ficheiros, que só pode ser eliminada com compactação
    - Ex: removeu-se **a**, **c** e **e**; um ficheiro d que ocupa 4 blocos só pode ser colocado após uma compactação...



  - Este sistema é útil em CD-ROMs e DVD-ROMs
    - conhece-se à partida a dimensão de cada ficheiro
    - não são feitas remoções de ficheiros após a gravação

Implementação do Sistema de Ficheiros

- Lista ligada de blocos
  - Mantém-se uma lista ligada dos blocos ocupados por cada ficheiro;
    Em cada bloco, para além dos dados, guarda-se também um apontador para o bloco seguinte do mesmo ficheiro
  - Vantagens:
    - Todos os blocos podem ser ocupados (a fragmentação não é problemática)
    - Ao SO basta saber a localização do 1º bloco.



Implementação do Sistema de Ficheiros

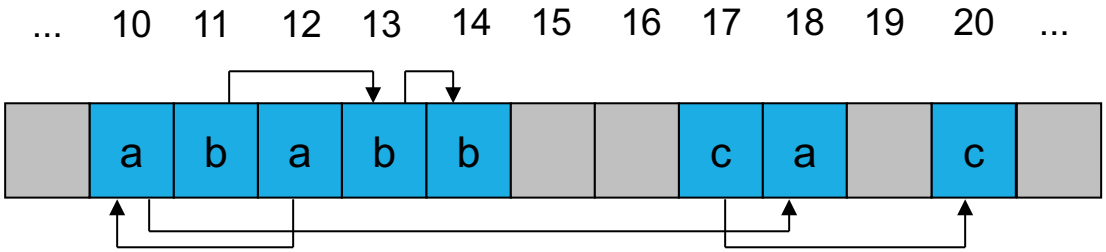
- Lista ligada de blocos
  - Desvantagens
    - Acesso sequencial – para chegar a um bloco é preciso passar pelos anteriores
      - Em ficheiros que ocupem muitos blocos espalhados pela partição o acesso aos últimos blocos é demasiado lento
      - Ex: para aceder ao último bloco de um ficheiro com 1000 blocos – será necessário que sejam lidos os 999 blocos anteriores
    - O tamanho real de cada bloco é diminuído pelo espaço ocupado pelo apontador

Implementação do Sistema de Ficheiros

- FAT – *File Allocation Table*
  - Manter em memória uma tabela com uma representação da lista ligada de blocos. Em cada posição da tabela indica-se o bloco seguinte do ficheiro.
  - Vantagens
    - Tal como no modelo anterior, a fragmentação não é problemática
    - Cada bloco é utilizado integralmente para armazenamento de dados (ao contrário do esquema anterior)
    - Facilita o acesso direto – para obter um bloco basta percorrer a FAT (mais rápido, pois percorre-se a memória e não o disco)
  - Desvantagem
    - A dimensão da FAT pode ser demasiado grande
      - Ex: 500GB de disco, blocos de 4KB indexados com 32bits (4Bytes)
        $\Rightarrow$ A FAT terá uma dimensão de 500MB

Implementação do Sistema de Ficheiros

- FAT – *File Allocation Table*
  - Exemplo:



No directório basta guardar o bloco de início de cada ficheiro

Ficheiro	1º bloco
a	12
b	11
c	17

FAT

...	
10	18
11	13
12	10
13	14
14	-1
15	0
16	0
17	20
18	-1
19	0
20	-1
...	

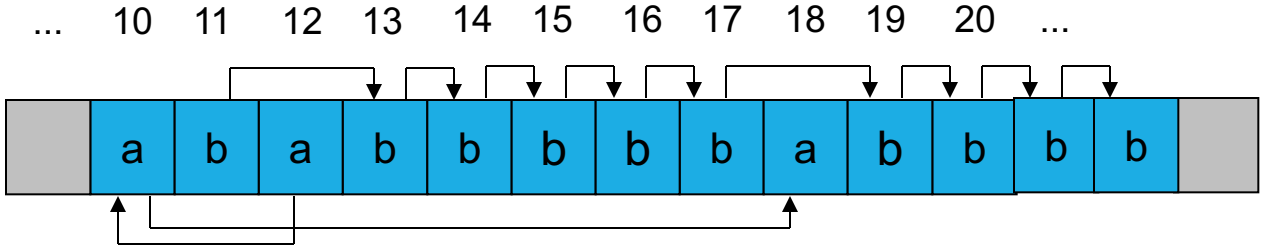
"0" representa bloco livre
" –1" representa último bloco do ficheiro

Implementação do Sistema de Ficheiros

- I-Nodes
  - Associa a cada ficheiro uma estrutura de dados contendo a localização em disco e os atributos do ficheiro
  - O i-node contém um número limitado de blocos do ficheiro
  - Para ficheiros de maior dimensão são atribuídos ao i-node outros blocos que contém tabelas com nºs de bloco extra
  - O i-node contém todas as características do ficheiro, exceto o nome que figura no (ou nos) diretórios onde o i-node é incluído
  - Vantagens
    - A fragmentação não é problemática
    - Para aceder a um ficheiro basta ter o respetivo i-node em memória (não é necessário dispor de toda uma tabela de alocação)
    - Facilita a partilha de ficheiros através de hard links

Implementação do Sistema de Ficheiros

- Exemplo:



I-node de *a*

Atributos de *a*
12
10
18
-
-
-
-
outros blocos: -

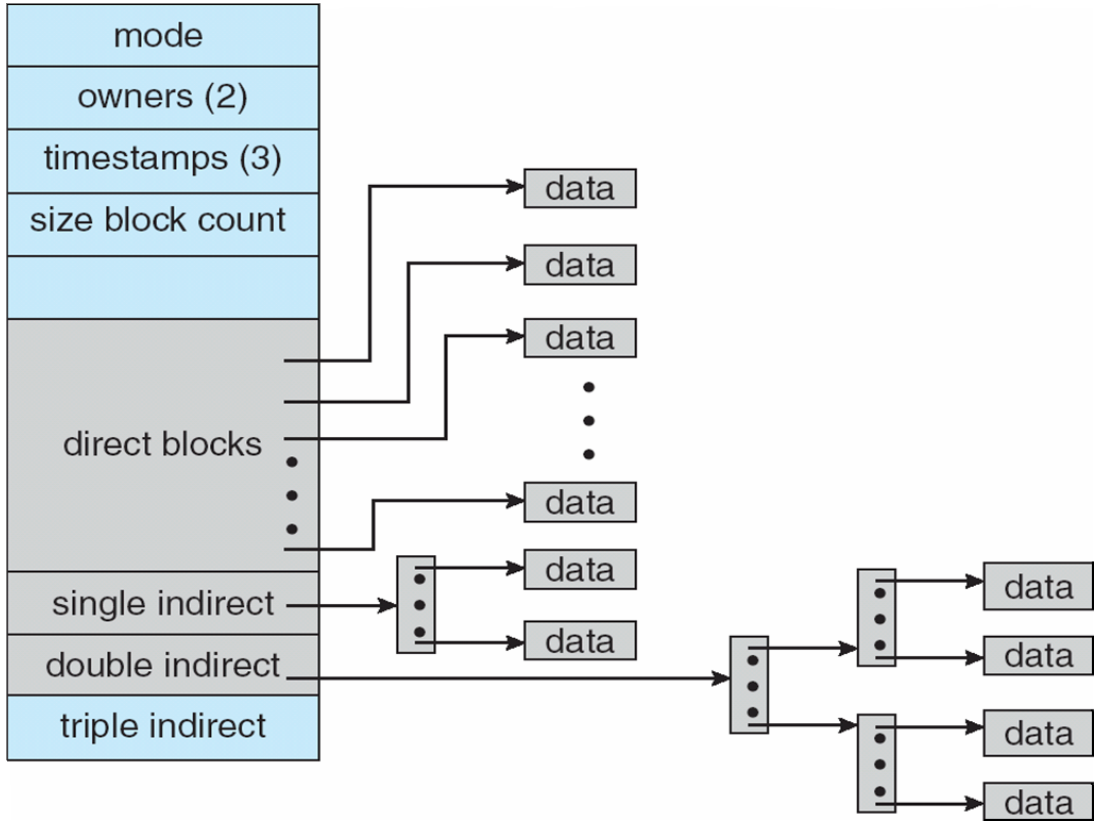
I-node de *b*

Atributos de *b*
11
13
14
15
16
17
19
outros blocos:
outros blocos
outros blocos

Blocos extra

20
21
-
-
-
-
-
-

I-nodes


mode
owners (2)
timestamps (3)
size block count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data

Implementação do Sistema de Ficheiros

- Implementação dos diretórios
  - Diretório
    - Um diretório é basicamente uma lista de nomes, a cada um dos quais se associam os respetivos atributos e localização em disco
    - O diretório é um ficheiro especial
  - Situações típicas
    - O diretório contém os atributos do ficheiro e a localização do primeiro bloco em disco - a partir do primeiro bloco localizam-se os restantes (ex: lista ligada, FAT,..)
    - O diretório contém o nome do ficheiro e o endereço de uma estrutura que contém os atributos do ficheiro e sua localização do em disco (ex: i-nodes)
  - Questões de implementação
    - Lidar com nomes de dimensão variável (fragmentação e compactação nos diretórios)
    - Procurar ficheiros em diretórios grandes (utilização de hash-tables e estruturas em árvore)

Implementação do Sistema de Ficheiros

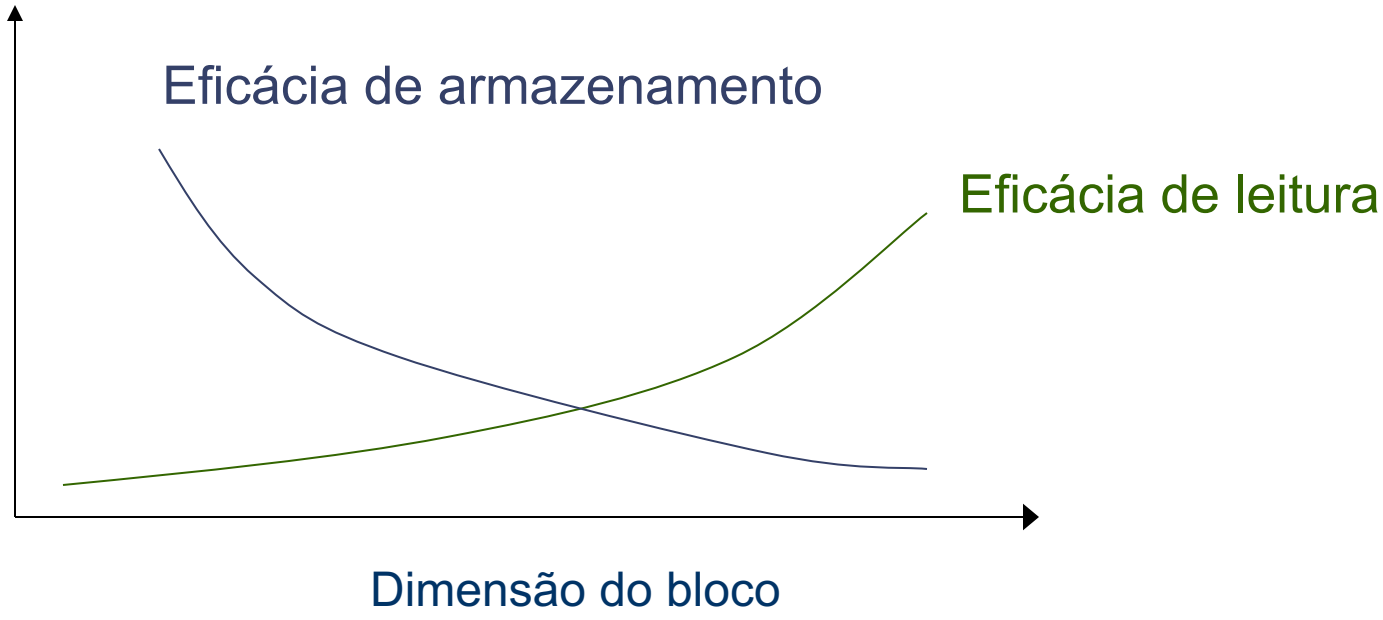
- Links
  - Possibilidade de um mesmo ficheiro figurar no sistema de ficheiros com vários nomes
  - *Hard link* (ligação real)
    - Incluir o mesmo ficheiro em mais que um diretório
    - Replicar em cada diretório os atributos e localização no disco
    - Implementação simples com i-nodes – basta replicar o nº de i-node
    - Implementação complexa se os atributos estiverem contidos no diretório – as alterações têm que ser replicadas em cada ligação
  - *Soft link* (ligação simbólica)
    - Incluir num diretório o nome de outro ficheiro que contém o caminho para o ficheiro original
    - Através do caminho acede-se à entrada de diretório do ficheiro original e, por essa via, aos seus atributos e localização em disco

Questões de Implementação

- Dimensão do bloco
  Qual será a melhor dimensão para os blocos?
  - Eficácia de leitura – relação entre o tempo de leitura e a informação efetivamente obtida do disco
    - Aumenta com a dimensão do bloco
      - menor *overhead* de posicionamento em cada leitura
      - menor número de leituras necessárias para obter os dados
  - Eficácia de ocupação – relação entre o espaço físico ocupado e o respetivo aproveitamento em termos de dados
    - Diminui com o aumento da dimensão do bloco
      - desperdício devido ao ajuste da dimensão do ficheiro para um número fixo de blocos
        exemplo: um ficheiro de dimensão 1 byte desperdiça o resto da dimensão do bloco

Questões de Implementação

- Variação da eficácia com a dimensão do bloco



Questões de Implementação

- Controlo da lista de blocos livres
  - Estrutura de dados de controlo dos blocos de disco não ocupados por ficheiros
  - Lista ligada
    - Lista ligada com o número de cada um dos blocos livres
    - Criação de um ficheiro:
      - Obter os primeiros blocos da lista (até à dimensão do ficheiro) e de seguida retirá-los da lista
    - Remoção de um ficheiro:
      - Acrescentar os respetivos blocos à lista
    - Método simples, mas poderá levar a dispersão dos ficheiros por vários blocos não contíguos
    - A dimensão da lista poderá ser bastante grande se o disco estiver pouco ocupado

Questões de Implementação

- Bitmap
  - Sequência com um número de bits igual ao número de blocos
    - Bit a “0” significa bloco livre
    - Bit a “1” significa bloco ocupado
  - Dimensão fixa
    - Quase sempre uma dimensão menor que na solução com lista (exceto quando disco está quase cheio)
  - Facilita a procura de blocos contíguos / próximos
    - Basta procurar 0's contíguos / próximos no bitmap

Questões de Implementação

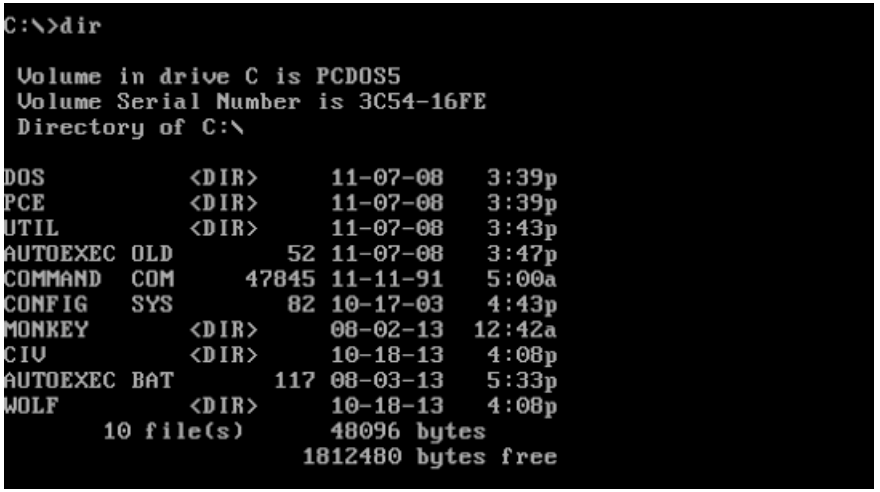
- Backup: Salvaguarda de segurança
  - Backup incremental – apenas as alterações desde o backup anterior
  - Com base no dispositivo físico – backup completo (imagem) de um disco
  - Com base na organização – backup de parte dos ficheiros
    - Evitam-se geralmente backups de:
      - Programas (pois podem-se reinstalar)
      - Ficheiros que modelizam dispositivos (bloco, caractere)
- Consistência
  - Mecanismos de verificação e recuperação da consistência do sistema de ficheiros

Questões de Implementação

- *Caching*
  - Manter em memória blocos mais recentes / maior probabilidade de uso futuro
- Leitura antecipada
  - Leitura e *caching*, por antecipação, de vários blocos
- Armazenamento contíguo
  - Tentar colocar o ficheiro em blocos de disco contíguos
  - Reduz-se o tempo de *overhead* relativo ao posicionamento no disco

MS-DOS e Windows 98 – FATs

- Características importantes
  - Sistema baseado em FAT
    - DOS: FAT-12 e FAT-16
    - Windows 95 e 98: FAT-32
  - Case-insensitive
  - Dimensão dos nomes
    - DOS: 8 (nome) + 3 (extensão) caracteres ASCII
      - O ficheiro é guardado com letras maiúsculas
    - Windows: 260 caracteres Unicode (2 bytes)



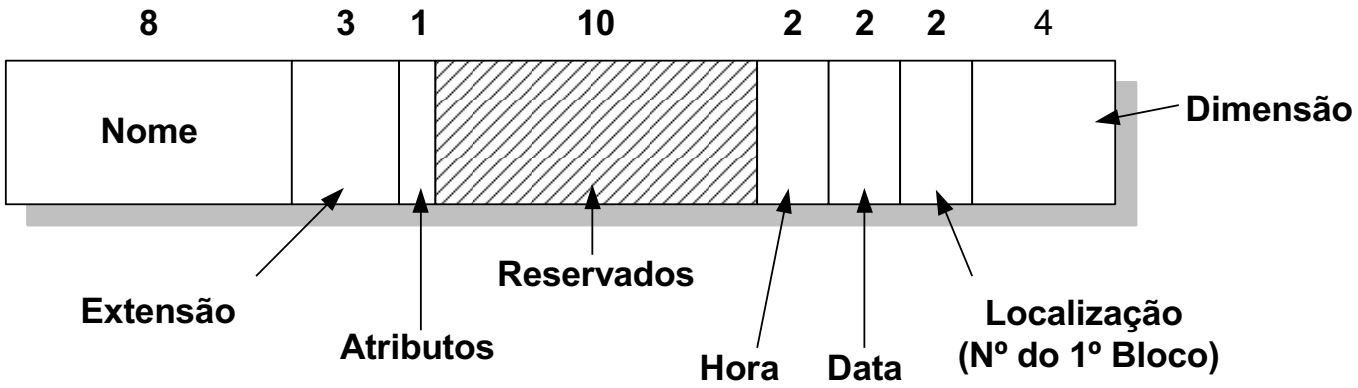
MS-DOS e Windows 98 – FATs

- Máxima dimensão do espaço indexável em disco

Bloco (bytes)	FAT-12	FAT-16	FAT-32
0.5K	2MB	X	X
1K	4MB	X	X
2K	8MB	128MB	X
4K	16MB	256MB	1TB
8K	X	512MB	2TB
16K	X	1GB	2TB
32K	X	2GB	2TB

MS-DOS e Windows 98 – FATs

- DOS – Directórios
  - Os directórios são ficheiros compostos por vários registos – um registo por cada ficheiro existente no directório
  - Cada registo tem o seguinte formato:

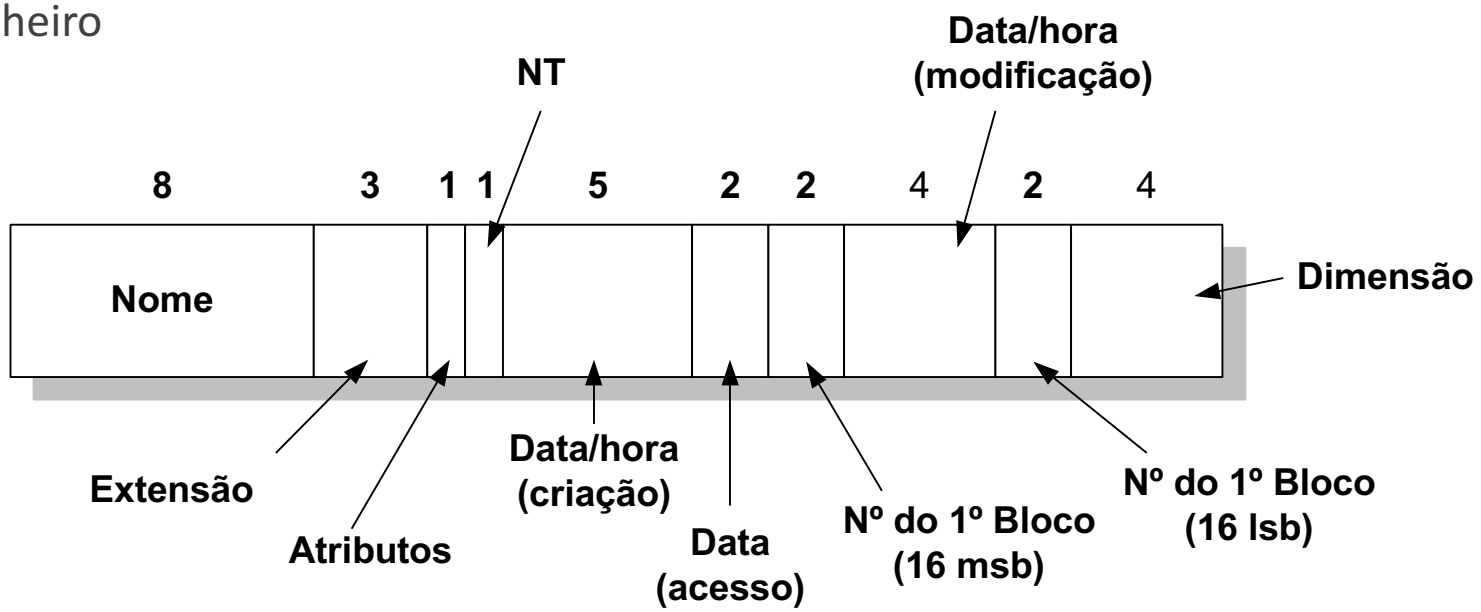


MS-DOS e Windows 98 – FATs

- DOS – Directórios
  - Nome e Extensão (8 bytes + 3 bytes)
    - Nome do ficheiro e extensão - dimensões fixas
  - Atributos (1 byte)
    - flags: hide, system file, read-only,...
  - Data/hora (2 + 2 bytes)
    - Data e hora da última modificação do ficheiro
  - Localização
    - Bloco inicial de localização do ficheiro => entrada na FAT;
  - Dimensão do ficheiro (4 bytes)
    - Dimensão real do ficheiro (em bytes)

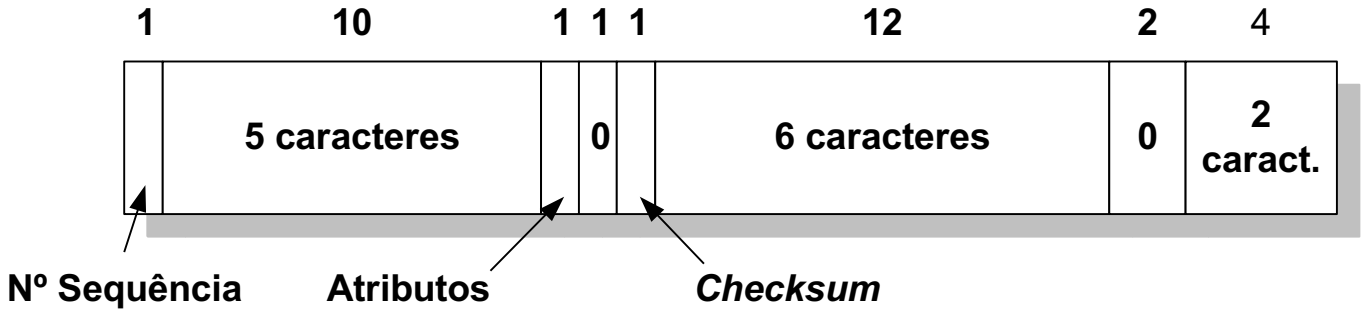
MS-DOS e Windows 98 – FATs

- Windows 98 – Directórios
  - Aproveitam-se os 10 bytes não utilizados pelo DOS
    - Introduzem-se mais campos para datas/horas e compatibilidade de nomes com Windows NT
    - Mais 2 bytes para obter os bits mais significativos do bloco de disco onde começa o ficheiro



MS-DOS e Windows 98 – FATs

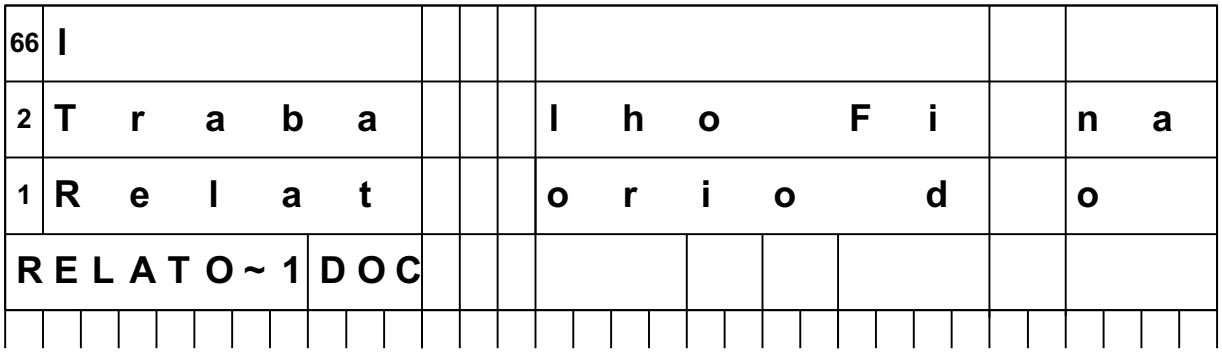
- Nomes compridos
  - A seguir a um registo com o formato anterior, podem-se seguir um conjunto de registos com o seguinte formato:



  - Deste modo o Windows consegue suportar nomes compridos
  - Para o DOS, entradas com este formato são ignoradas, pois contêm um valor de atributo inválido para o DOS

MS-DOS e Windows 98 – FATs

- Nomes compridos
  - Exemplo:
    Ficheiro chamado “Relatorio do Trabalho Final”



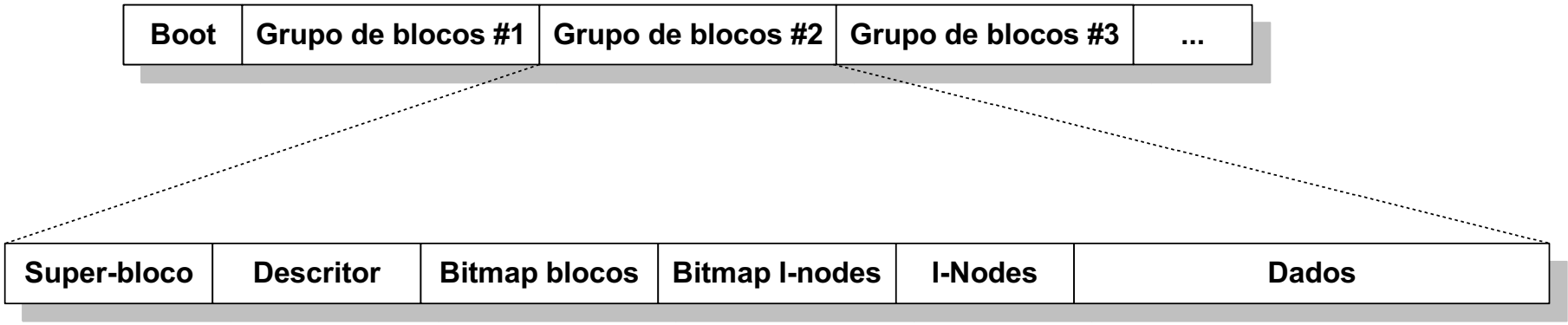
  - O DOS só consegue lê a primeira entrada
  - O Windows 98 utiliza todas as que se seguem até encontrar um número de sequência com um desfasamento de 64 unidades

Linux – ext2

- Características importantes
  - Sistema de ficheiros nativo do Linux
    - Semelhante ao sistema BFF (Berkeley Fast File system)
  - Sistema de ficheiros baseado em I-Nodes
    - Tal como todos os sistemas de ficheiros Unix
  - Nomes de ficheiros até 255 caracteres
  - Case-sensitive

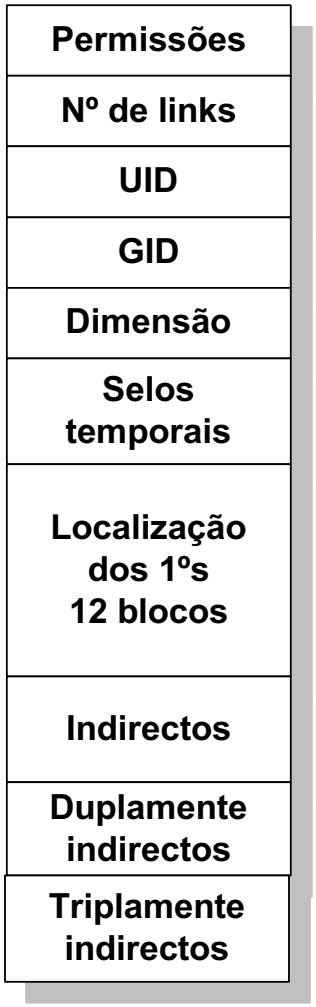
Linux – ext2

- Estrutura de uma partição
  - Cada partição é dividida em vários grupos de blocos, tirando-se partido da arquitectura física do disco
  - Cada ficheiro reside preferencialmente num único grupo de blocos
  - Deste modo acelera-se a leitura/escrita no disco



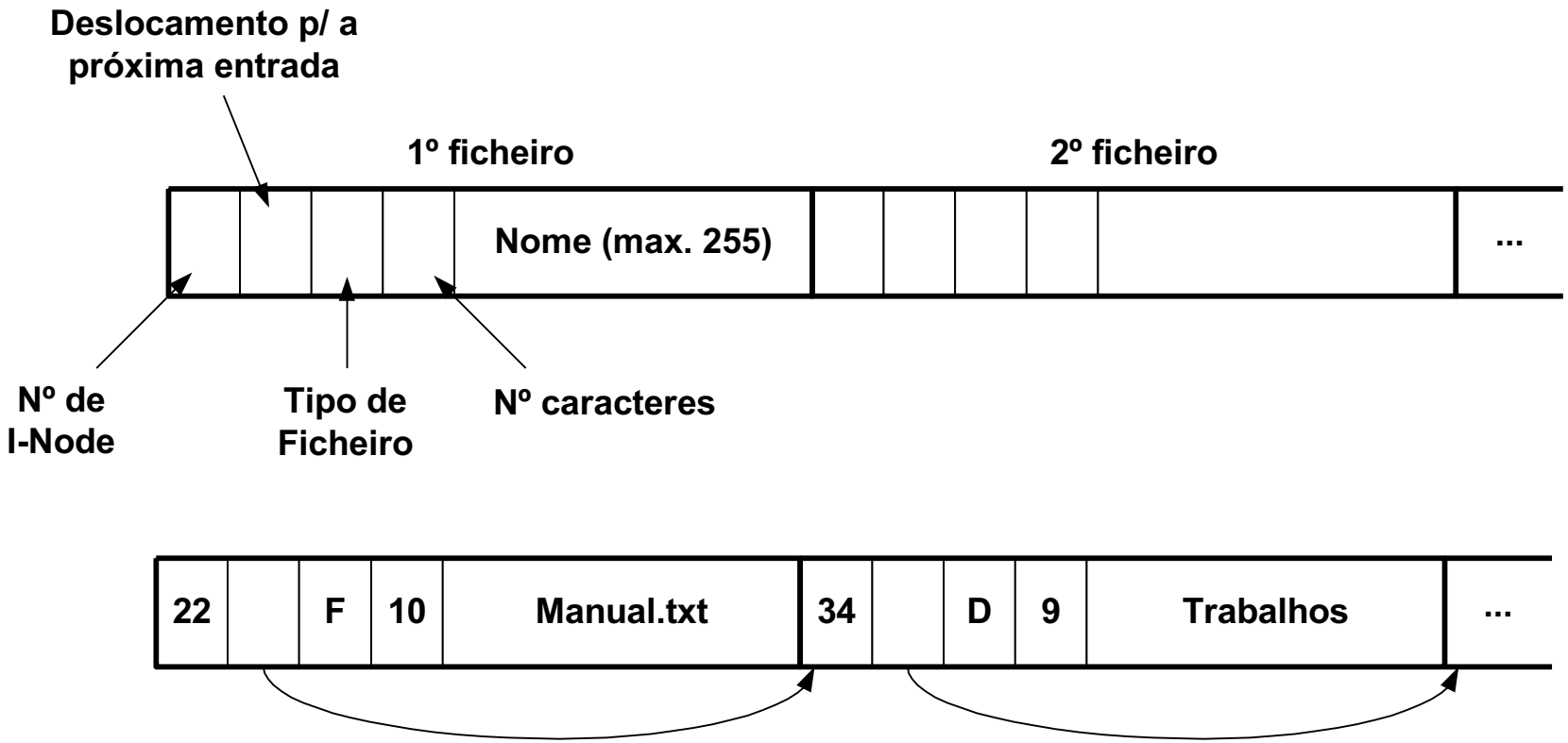
Linux – ext2

- Estrutura dos I-Nodes
  - Dimensão de 128 bytes
  - Os últimos 3 campos são também endereços de blocos do disco:
    - Indiretos – endereço de um bloco do disco que contém mais endereços de blocos da localização do ficheiro
    - Duplamente indiretos – endereço de uma tabela de blocos indiretos
    - Triplamente indiretos – endereço de uma tabela de blocos duplamente indiretos
- (O objetivo é permitir que um ficheiro esteja localizado em mais de 12 blocos)



Linux – ext2

- Estrutura dos directórios



TPC

- Ler os seguintes artigos
  - http://arstechnica.com/gadgets/2008/03/past-present-future-file-systems/1/