O SISTEMA DE FICHEIROS

Ficheiro

- **programa ou conjunto de dados gravados em memória secundária e identificado por um nome**
- **conjunto de dados persistentes, geralmente relacionados entre si e identificados por um nome**

Sistema de gestão de ficheiros / Sistema de ficheiros

- **parte do sistema operativo responsável pelos serviços de**
  - » **manipulação de ficheiros (criação, destruição, escrita, leitura, cópia, …)**
  - » **controlo do acesso aos ficheiros por parte dos processos e dos utilizadores**
  - » **gestão dos recursos associados aos ficheiros (blocos em disco, directórios, ...) garantindo a sua fiabilidade**

Interface do Sistema de Ficheiros

Ficheiros

- **Tipos**
  - » **dados (numéricos, caracteres, …)**
  - » **programas (fonte, objecto)**

  Indicar o tipo, para evitar operações erradas:
  - » **como parte do nome**
  - » **no conteúdo do próprio ficheiro (ex: indicação da aplicação com que foi criado)**

- **Atributos**
  - » **nome, tipo, localização, tamanho, data, dono, protecções de acesso, …**
  - » **Mantidos na estrutura do directório ou em estruturas associadas aos ficheiros (ex: num *i-node,* UNIX)**

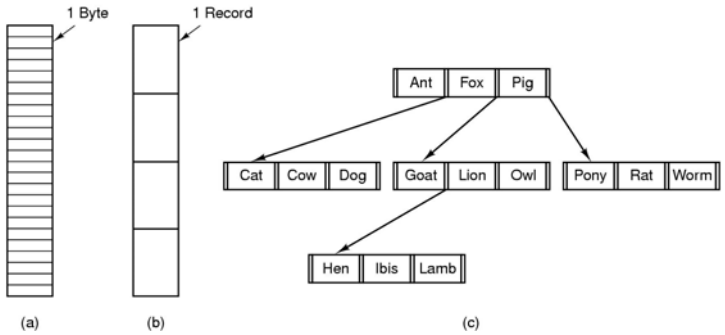
- **Operações**
  - » **criação, abertura, fecho, leitura, escrita, posicionamento do apontador do ficheiro, ...**

Interface do Sistema de Ficheiros

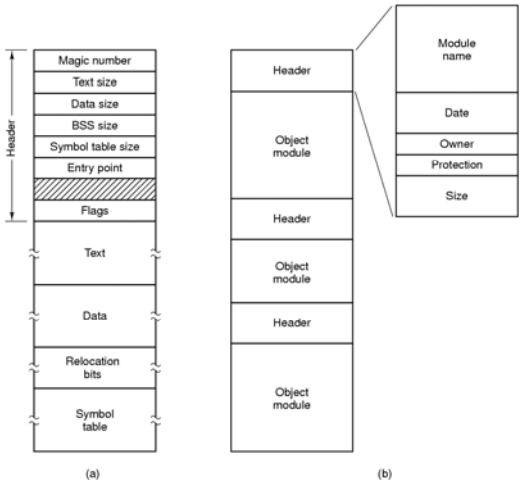
Ficheiros (cont.)

- **Tabela de ficheiros abertos**
  - » **mantida em memória (acesso rápido)**
  - » **Open - criar entrada na tabela; retorna índice/apontador p/ essa entrada**
  - » **Close - retirar elemento da tabela**
  - » **em UNIX: uma tabela de ficheiros abertos por processo + uma tabela de ficheiros abertos global**
- **Estrutura de um ficheiro**
  - » **Vista pelo utilizador**
    - **colecção de *bytes* (ex: UNIX)**
    - **colecção de registos (comprimento fixo ou variável)**
    - **estrutura complexa (ex: documento formatado, programa executável)**
  - » **Vista pelo sistema operativo**
    - **colecção de blocos**
      - – **bloco - unidade de transferência física entre o disco e a memória**
        1 bloco = N sectores

ESTRUTURAS DE DADOS DO S.O. RELATIVAS AO SISTEMA DE FICHEIROS



Estrutura de um ficheiro vista pelo utilizador
a) colecção de bytes (ex: UNIX)
b) colecção de registos (comprimento fixo ou variável)
c) estrutura complexa (ex: árvore)



Tipos de ficheiros: a) executável; b) arquivo

Interface do Sistema de Ficheiros

Ficheiros (cont.)

- **Métodos de acesso**
  - » **Sequencial**
  - » **Directo / aleatório**
    - ⇒ **disco + registos de comprimento fixo**
  - » **Outros (indexado, sequencial indexado, …)**

- **Facilidades fornecidas por alguns sistemas operativos**
  - » **Partilhar secções de um ficheiro entre vários processos.**
  - » **Fazer o *lock* de secções de um ficheiro aberto, acedido p/ vários processos.**
  - » **Mapear secções de um ficheiro na memória.**

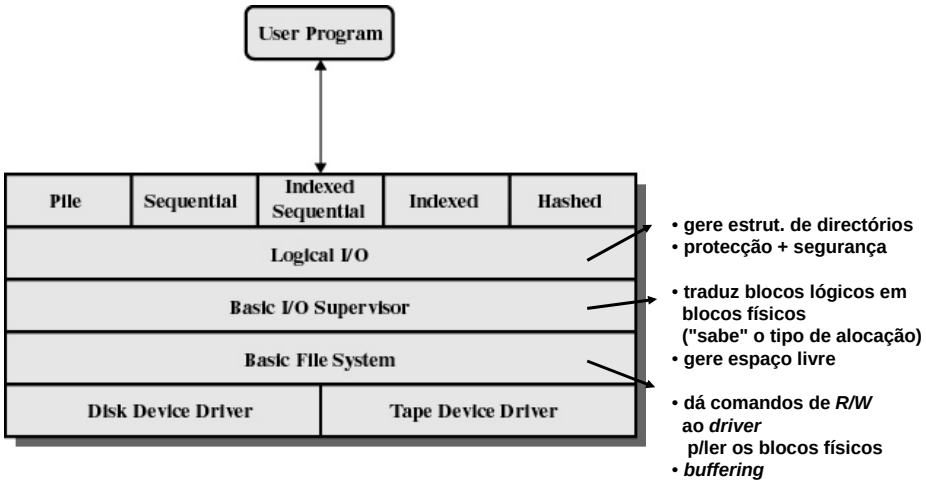
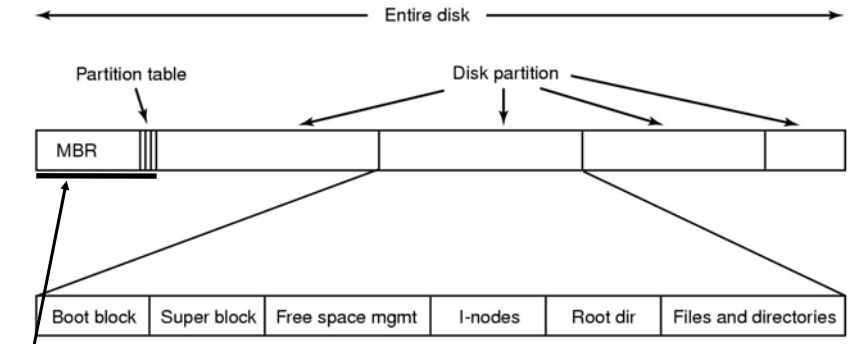


Figure 12.1 File System Software Architecture [GROS86]

Interface do Sistema de Ficheiros

Partições e Directórios

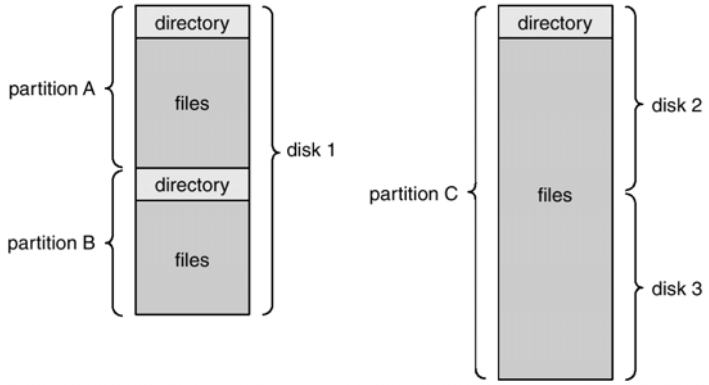
- Necessidade de aceder aos ficheiros por nome
  - » o directório associa nomes de ficheiros a índices do mapa de ficheiros
  - » um directório é um ficheiro especial
- Necessidade de organização dos ficheiros
  - » partições do disco
  - » directórios
    - eficiência - localização rápida dos ficheiros
    - conveniência - possibilidade de haver 2 ficheiros c/ o mesmo nome (em directórios diferentes)
    - organização - agrupamento lógico dos ficheiros de acordo c/ as suas propriedades ou tipo
- Operações sobre directórios
  - » procurar ficheiro, criar / destruir ficheiro, listar, ...
- Estrutura dos directórios
  - » único nível
  - » em árvore
  - » grafo acíclico
  - » grafo genérico



Estrutura possível de um sistema de ficheiros (UNIX-like)

mais recentemente GPT - GUID Partition Table

UNIX		WINDOWS NTFS
boot block	⟷	*partition boot sector*
super block	⟷	*Master File Table (MFT)*
inode	⟷	*File Control Block (FCB)*

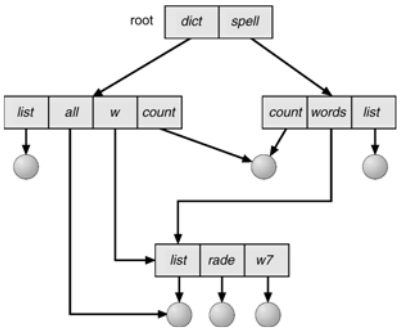


Organizações possíveis de sistemas de ficheiros

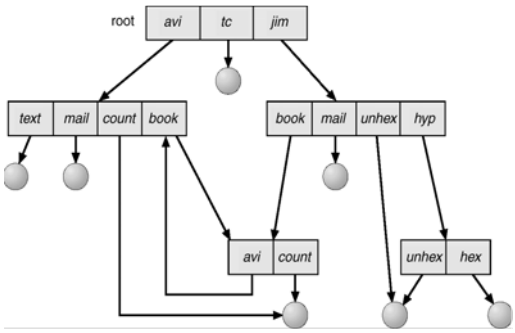
Interface do Sistema de Ficheiros

Directórios (cont.)

- **Directório c/ único nível**
  - » **Todos os ficheiros no mesmo directório.**
  - » **Problemas:**
    - **nomes únicos**
    - **tempo de pesquisa longo**
- **Directório c/ estrutura em árvore**
  - » **Estrutura mais comum.**
  - » **Um directório pode conter ficheiros ou outros directórios (subdirectórios).**
  - » **Usar um caminho (absoluto/relativo) p/ especificar um ficheiro.**
- **Directório em forma de grafo acíclico**
  - » **Generalização da estrutura em árvore.**
  - » **Permite partilha de ficheiros/directórios (várias entradas de diferentes directórios a apontarem p/ a mesma posição do disco)**
  - » **Um ficheiro pode ter vários *pathnames*.**



Directório em forma de grafo acíclico



Directório em forma de grafo genérico

Interface do Sistema de Ficheiros

- ***hard links*** **- várias entradas de directórios referenciam a mesma entrada do mapa de ficheiros (*inode, UNIX*)**
  - » **ex: em UNIX →**
  - » `ln existing_file new_file`
  - » **útil p/ substituir referência longo p/ referência curta**
    ex: `ln /users/silva/progs/prob_15.c prob_15`
  - » **não é criado nenhum *inode* novo**
    `/users/silva/progs/prob_15.c` **e** `prob_15` **referenciam o mesmo *inode***
  - » **só o *superuser* pode criar *hard links* p/ directórios p/ evitar ciclos**
- ***soft links*** **- as entradas do directório contêm *pathnames* (útil p/ referenciar ficheiros noutros *file systems*; não incrementa a *reference count*)**
  - » **ex: em UNIX →**
  - » `ln -s existing_file new_file`
- **Evitam duplicações de dados.**
- **Facilitam a partilha de dados.**

Interface do Sistema de Ficheiros

Protecção

Proteger contra
- dano físico → fiabilidade
- acesso impróprio → protecção propriamente dita

Proteger ⇒ controlar
- o que pode ser feito sobre o ficheiro/directório (*Read, Write, Delete, List, …)*
- por quem

Listas de acesso (ex: Multics, Windows NT e seguintes)
- uma lista associada a cada ficheiro/directório indicando o *username* e o tipo de acesso
- problemas: comprimento da lista; tamanho variável

Grupos de utilizadores (ex: UNIX)
- dividir os utilizadores em classes: dono, grupo, outros
- o gestor do sistema cria os grupos e associa os utilizadores a um grupo
- para cada ficheiro definir o tipo de acesso permitido (*Read, Write, Execute*) para cada classe de utilizadores

Passwords
- associar uma *password* a cada ficheiro / directório

Implementação do Sistema de Ficheiros

Estrutura do sistema de ficheiros

- ***I/O*** **entre memória e disco - feita em blocos p/ melhorar a eficiência**
  - » 1 bloco = N sectores
- **Estruturação do sistema de ficheiros em níveis**
  - » ***(Aplicações)***
  - » **Sistema de ficheiros lógico (*logical file system*)**
    - Usa a estrutura de directórios p/ fornecer ao módulo seguinte o que ele precisa, dado o nome de um ficheiro.
    - Responsável pela protecção e segurança.
  - » **Módulo de organização de ficheiros**
    - Conhece os blocos lógicos e físicos que compõem um ficheiro.
    - Traduz os endereços de bloco lógicos em endereços de bloco físicos.
    - Faz a gestão do espaço livre.
  - » **Sistema de ficheiros básico (*basic file system*)**
    - Dá comandos ao *device driver* p/ ler ou escrever blocos físicos do disco (endereço do bloco = nº do *drive*, nº do cilindro, nº da pista, nº do sector).
  - » **Controlo de *I/O***
    - *device drivers* e *interrupt handlers* p/ transferir informação entre mem. e disco
  - » ***(Dispositivos)***

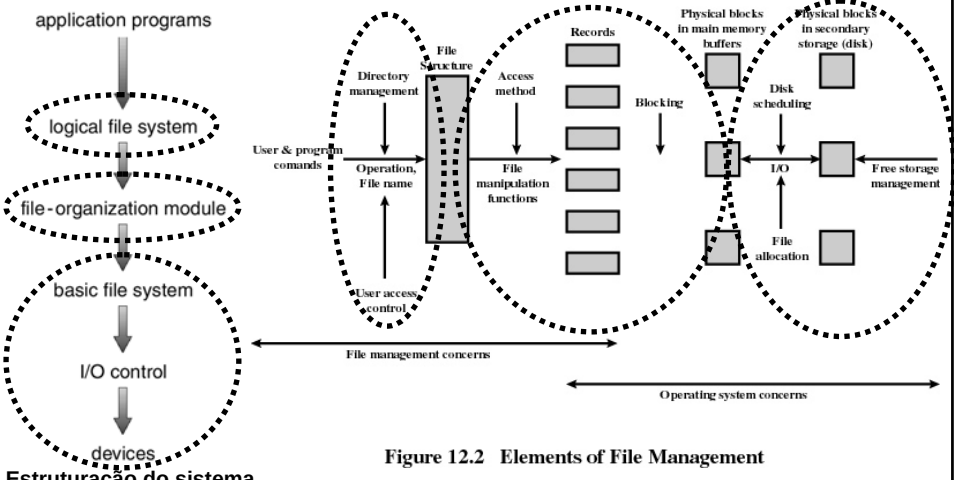


Figure 12.2 Elements of File Management

Estruturação do sistema de ficheiros em níveis

Implementação do Sistema de Ficheiros

Estrutura do sistema de ficheiros (cont.)

- **Tabela de ficheiros abertos**
  - » **mantida em memória pelo sistema operativo**
  - » **contém informação sobre os ficheiros actualmente abertos**
- **Montagem *(mount)* de um sistema de ficheiros**
  - » **o sistema de ficheiros tem de ser montado (associado a um directório) antes de poder ser usado**
  - » **em alguns sistemas, esta operação é feita automaticamente, sempre que o sistema encontra um disco**

Métodos de alocação

- **Como alocar espaço p/ os ficheiros, de modo a**
  - » **usar o espaço do disco eficientemente**
  - » **aceder aos ficheiros rapidamente**
- **Vários métodos (em geral, apenas um é suportado)**
  - » **alocação contígua**
  - » **alocação ligada**
  - » **alocação indexada**

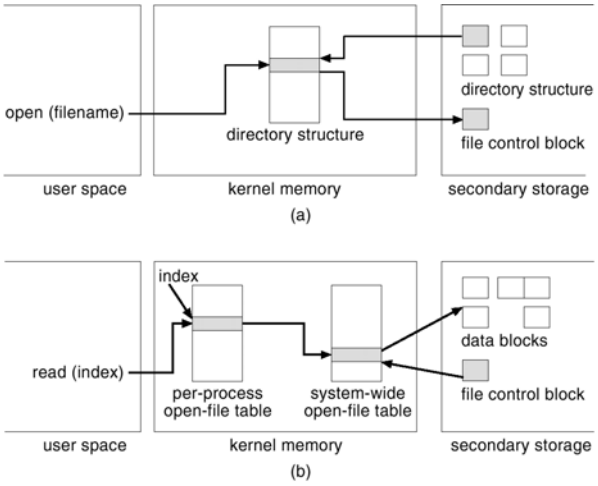
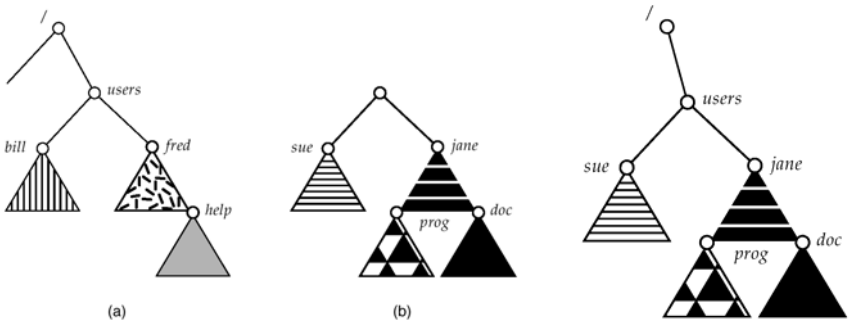


Tabela de ficheiros abertos

Montagem (*mount*) de um sistema de ficheiros



Sistemas de ficheiros separados

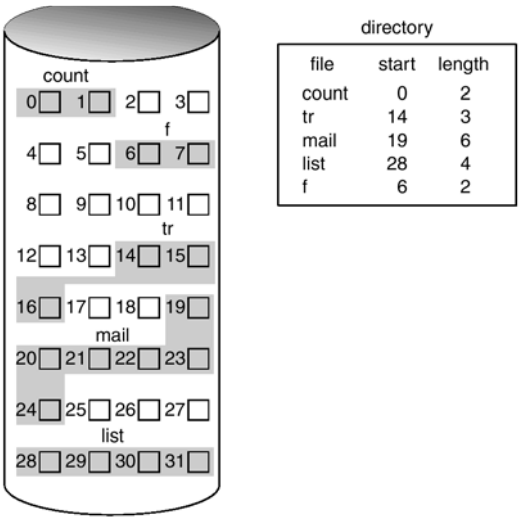
Depois de montagem

Métodos de alocação
Alocação contígua

- **Cada ficheiro é armazenado num conjunto de blocos contíguos do disco.**

- **A entrada do directório p/ cada ficheiro deve indicar**
  - **o endereço do bloco inicial**
  - **o número de blocos**

Vantagens
- **simples**
- **fácil acesso (sequencial / directo)**
- **poucos posicionamentos (*seeks*) p/ aceder ao ficheiro**
- **eficiente em aplicações que processam o ficheiro completo**



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Alocação contígua

Alocação contígua

Dificuldades

- **Indicar o tamanho final do ficheiro quando ele é criado.**
- **Tentação p/ alocar demasiado espaço → desperdício.**
- **Encontrar espaço livre c/ a dimensão necessária.**
- **Dificuldade de crescimento**
  - » **encontrar espaço vago maior e copiar o ficheiro para lá;**
  - » **alocação contígua modificada : juntar uma extensão ao ficheiro**
- **Apagamento de dados pouco eficiente.**
- **Fragmentação externa ⇒ necessidade de compactação do espaço livre.**

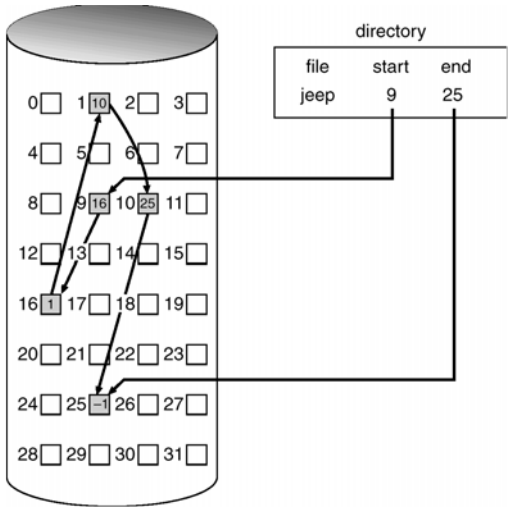
Alocação ligada

Resolve todas as dificuldades da alocação contígua, mas tem outros problemas.

- **Cada ficheiro é uma lista ligada de blocos**
  - **cada bloco contém (informação + um apontador p/ o bloco seguinte)**
- **Os blocos podem estar dispersos pelo disco.**
- **O directório tem apontadores p/ o primeiro e o último bloco do ficheiro.**

Vantagens

- **Os ficheiros podem crescer dinamicamente.**
- **É fácil inserir / eliminar blocos intermédios.**
- **Não há fragmentação externa.**
- **É fácil gerir o espaço livre.**



Alocação ligada

Alocação ligada

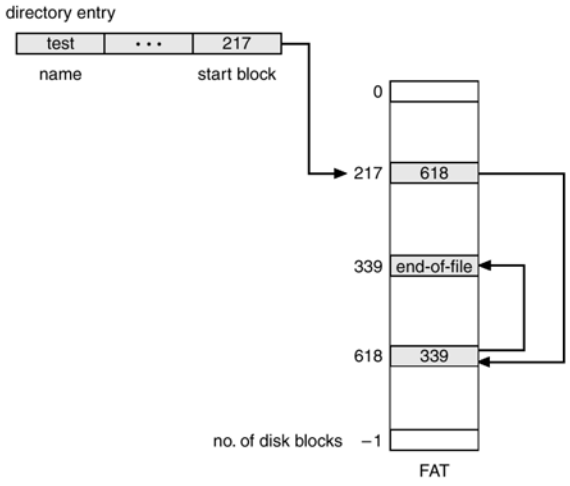
Dificuldades

- **O acesso directo é "pesado".**
- **Os apontadores consomem espaço de disco**
  - » **reduzir este espaço ⇒ alocar *clusters* de blocos (1 *cluster* = N blocos) mas... aumenta a fragmentação interna**
- **Fiabilidade reduzida**
  - » **Perder um apontador ⇒ perder o resto do ficheiro / ligar o ficheiro a outro**
  - » **Solução parcial (⇒ mais espaço):**
    - **lista duplamente ligada**

Alocação ligada

Variante deste método

- **Usar uma *FAT (File Allocation Table)* para manter informação de quais os blocos (*clusters*) que fazem parte de um ficheiro.**
- **Também mantém informação de quais são os blocos que estão livres ou em mau estado.**
- **Esta tabela é constituída por uma entrada por cada bloco do disco.**
  - » **Cada entrada é indexada por um número de bloco e contém o endereço do bloco seguinte. O último bloco contém um valor especial (por ex: 1111111111...).**
  - » **O directório contém o nº do 1º bloco do ficheiro.**
- **Vantagens:**
  - » **Melhora o acesso aleatório.**
  - » **Reduz o número de posicionamentos (a *FAT* pode estar em memória principal).**



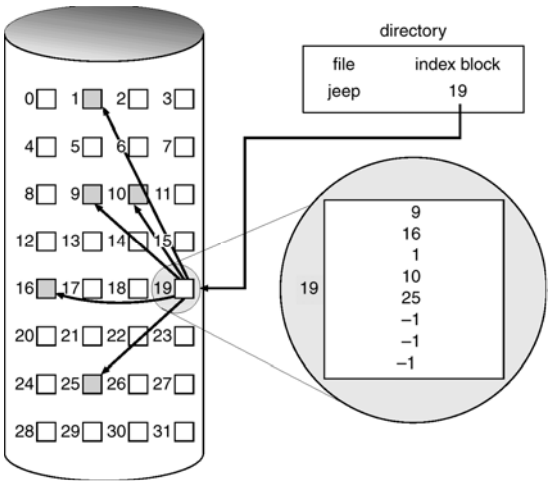
FAT - File Allocation Table

Alocação indexada

- Cada ficheiro tem uma tabela de índices c/ tantas entradas quantos os blocos do ficheiro.
- O elemento *i* da tabela contém o endereço do bloco *i* do ficheiro.
- O directório contém o endereço da tabela de índices de cada ficheiro.

Dificuldades

- Conhecer previamente o tamanho do ficheiro p/ determinar o tamanho da tabela de índices.
  - » Pode-se reservar, à partida, um bloco p/ a tabela de índices (bloco de índices) ⇒ o tamanho do ficheiro fica limitado
  - » Soluções p/ permitir o crescimento do ficheiro (v. adiante)
- Espaço ocupado pela tabela de índices.
- Nº de posicionamentos elevado.



Alocação indexada

Alocação indexada

Vantagens

- Facilidade de crescimento do ficheiro (até um máximo especificado).
- Acesso directo rápido.
- Evita a fragmentação externa.

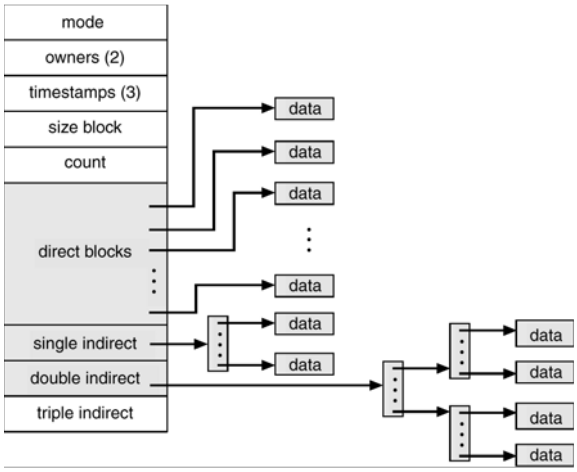
Soluções p/ permitir o crescimento do ficheiro

- Lista ligada de índices
  - » Vários blocos de índices, constituindo uma lista ligada.
- Índice multinível
  - » Usar um índice de 1º nível que aponta para um índice de 2º nível que aponta para os blocos de dados.
  - » Extensível para N níveis.
- Índice directo, combinado com índice multinível
  - » As primeiras entradas do bloco de índices apontam directamente p/ blocos de dados.
  - » As últimas entradas do bloco de índices apontam p/ outros blocos de índices.

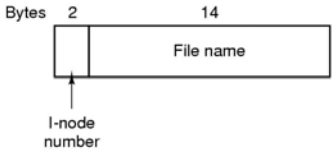
Alocação indexada

- Índice directo, combinado com índice multinível (cont.)
  - » Vantagens
    - Possibilidade de crescimento dos ficheiros. Existe um tamanho máximo que é suficiente p/ a maioria das aplicações.
    - Acesso rápido p/ ficheiros pequenos (acedidos usando apontadores directos)
  - » Dificuldades
    - Acesso a ficheiros grandes não é muito eficiente.
    - Muitos posicionamentos.
  - » Exemplo: *Inodes* do UNIX
    - Cada ficheiro, directório ou dispositivo de I/O tem um *Inode* associado.
    - Cada entrada do directório aponta p/ o *Inode* respectivo.
    - Cada *Inode* contém:
      - tipo de ficheiro: regular, directório, *block special*, *character special*
      - permissões de acesso
      - identificador do dono e do grupo
      - contagem de *hard links*
      - data e hora da última modificação
      - localização dos blocos (só p/ ficheiros regulares e directórios)
      - *major e minor device numbers* (só para ficheiros especiais)
      - valor do *symbolic link* (se for um *symbolic link*)

Índice directo, combinado com índice multinível



I-node de UNIX



Entrada de um directório

Performance dos Métodos de Alocação

Depende do tipo de acesso mais usado: sequencial ou directo.

Alocação contígua
- **boa p/ qualquer tipo de acesso**

Alocação ligada
- **boa p/ acesso sequencial**
- **má para acesso aleatório**

Alocação indexada
- **a performance depende de**
  - » **nº de níveis dos índices**
  - » **tamanho do ficheiro**
  - » **posição do bloco desejado**

Gestão do espaço livre

Técnicas:

- **tabela de *bits***
- **lista ligada de blocos livres**

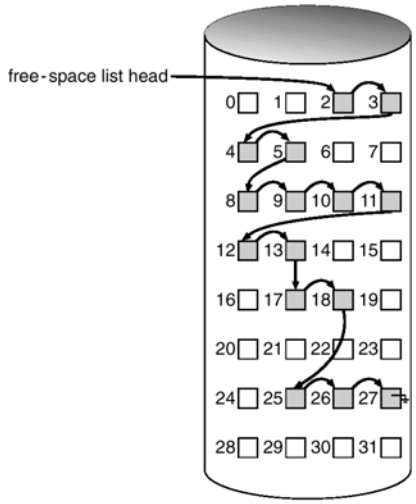
Tabela de *bits*

- **1 bloco ↔ 1 *bit*** **(ex: 1 ⇔ bloco ocupado; 0 ⇔ bloco livre)**
- **Vector de *bits* p/ representar todos os blocos do disco.**
- **Vantagens:**
  - » **simples de implementar**
  - » **ocupa pouco espaço; pode ser mantido em memória**
  - » **fácil encontrar o 1º bloco livre ou conjunto de blocos livres consecutivos**
- **Dificuldades**
  - » **o mapa de *bits* também tem de ser mantido no disco; as cópias em memória e no disco têm de ser consistentes.**

Gestão do espaço livre

Lista ligada de blocos livres

- **Ligar, através de apontadores, todos os blocos livres, mantendo numa posição especial do disco, um apontador p/ o 1º bloco livre.**
- **Vantagem**
  - » **não ocupa espaço**
- **Dificuldades**
  - » **travessia da lista pouco eficiente (operação rara, a menos que se queira encontrar n blocos livres consecutivos)**
- **Variante deste método**
  - » **O 1º bloco livre contém os endereços de *n* blocos livres.**
  - » **Os primeiros *n-1* endereços indicam blocos livres, de facto.**
  - » **O *n*-ésimo endereço indica um bloco livre c/ uma constituição semelhante a este**
  - » **Vantagem (relativamente à lista ligada simples):**
    - **os endereços de um grande nº de blocos livres podem ser encontrados rapidamente.**



Lista ligada de blocos livres

Implementação dos Directórios

Lista linear

- Cada elemento da lista deve conter pelo menos o nome do ficheiro e um apontador para os blocos de dados.
- Em alguns sistemas, alguma da informação associada ao ficheiro é guardada num cabeçalho do ficheiro.
- (-) O tempo de pesquisa necessário p/ encontrar um ficheiro pode ser grande.

Tabela de *hash*

- A tabela de *hash* tem como entrada um valor calculado a partir do nome do ficheiro e retorna um apontador p/ a entrada correspondente ao ficheiro, numa lista linear.
- Diminui o tempo de pesquisa de um ficheiro.
- Necessário tratar situações de colisão, quando 2 ficheiros dão origem ao mesmo índice.

Eficiência e Performance

Eficiência na utilização do disco depende de:

- **Algoritmos de alocação de espaço p/ os ficheiros e de manipulação de directórios**
  - » **ex: p/ reduzir fragmentação interna, usar 2 tamanhos de *clusters* :**
    (o último *cluster* de um ficheiro pode ser mais pequeno que os outros)
    4.2 BSD → blocos de 4KB e fragmentos de 1KB
    1 ficheiro c/ 18KB ocupa 2 blocos + 2 fragmentos
- **Tipos de dados mantidos em cada entrada do directório**
  - » **ex: data da última modificação → importante p/ os *backup*'s**
    mas... data da última leitura (ocupa tempo e espaço, necessário ?)
- **Tipo de apontadores usados para aceder aos dados**
  - » **Maior nº de *bits* ⇒**
    - **maior espaço consumido na manutenção de índices, listas ligadas, ...**
      (ex: métodos de alocação, gestão do espaço livre)
    - **mas ... possibilidade de usar discos maiores sem aumentar a fragmentação interna**
      Ex: FAT12, FAT16, FAT32

Performance

Algumas formas de a melhorar:

- ***cache* do controlador**
  - » **p/ manter uma pista do disco**
    a partir do sítio onde a cabeça ficou, após o posicionamento
- ***cache* de disco**
  - » **p/ manter blocos acedidos recentemente**
- **no acesso sequencial**
  - » **técnica de *free-behind***
    - **remover um bloco do *buffer* logo que se lê outro bloco do disco**
  - » **técnica de *read-ahead***
    - **ler o bloco pedido e alguns que se lhe seguem**
- **RAM disk ou disco virtual**
  - » **aceita as operações comuns sobre discos mas efectua-as sobre RAM**
  - » **⇒ *device driver* adequado**
  - » **útil p/ ficheiros temporários (ex: usados por um compilador)**

Recuperação

Verificação de consistência

- **correr um programa que compara os dados na estrutura de directórios c/ os dados nos blocos de dados e tenta determinar inconsistências.**
  - » **ex: `fsck` (file system check) do UNIX; `scandisk` do MS Windows**

Backup* e *Restore

- **ciclo de cópias integrais (todos os ficheiros) e incrementais (apenas os ficheiros alterados desde a última cópia) usando diferentes discos/fitas que permite recuperar qualquer ficheiro acidentalmente destruído durante o ciclo.**

Journaling

- **registar automaticamente as operações executadas sobre o sistema de ficheiros, mantendo um registo contínuo dessas operações num arquivo ("journal")**
- **após uma falha, este registo pode ser usado para repor o sistema de ficheiros num estado conhecido, consistente.**
- **sistemas de ficheiros que *journaling*: ext3, ext4, NTFS, e outros**