Sistemas Operativos
8ª parte - Gestão de Ficheiros

Prof. José Rogado
jrogado@ulusofona.pt
Prof. Pedro Gama
pedrogama@gmail.com
Universidade Lusófona

Gestão de Ficheiros

- Enquadramento
- Ponto de vista do utilizador
- Arquitectura de Informação
- Estruturas para a Gestão de Dados
- Gestão do espaço em disco
- Interface de Acesso
- Tipos de Ficheiros
- Exemplos de SGFs

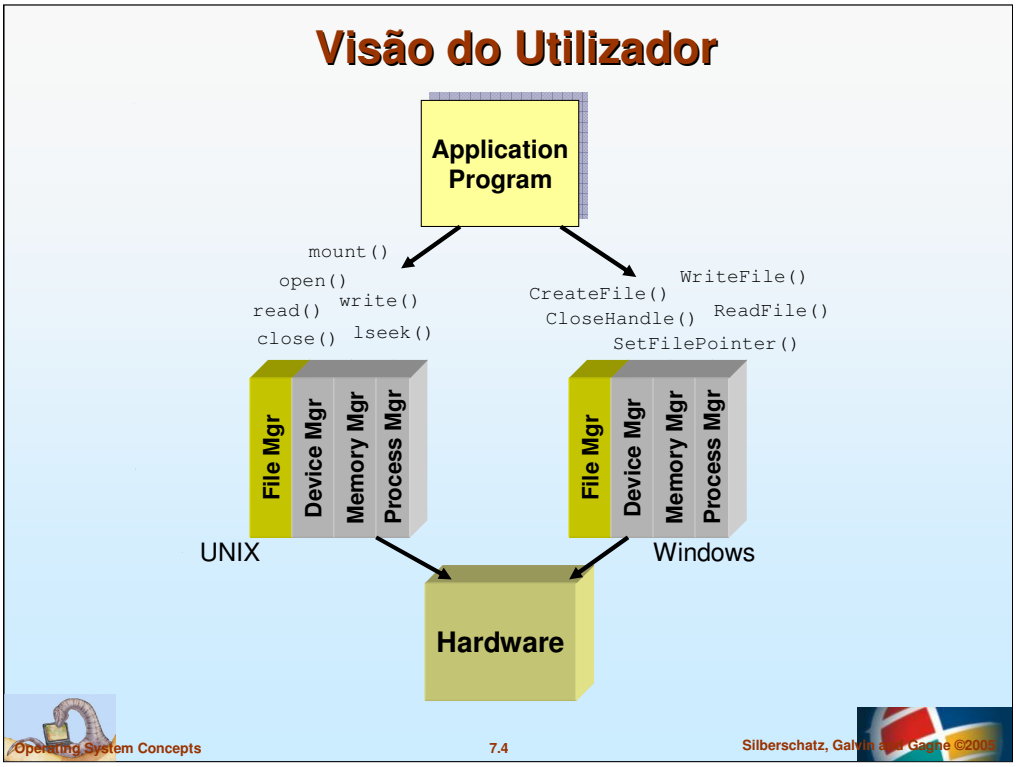
Objectivos para a LIG:

. perceber que o sistema de ficheiros é uma abstração oferecida pelo SO

. alguns problemas inerentes à gestão do espaço em dispositivos de armazenamento.

Requisitos da Gestão de Ficheiros

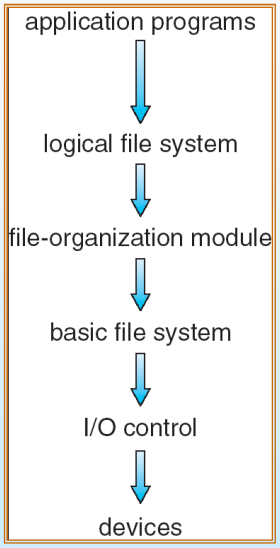
- O espaço disponível num meio de armazenamento é uma sequência de bytes sem estrutura
- Para armazenar informação nesse espaço são necessários alguns requisitos básicos
  - Distinguir o espaço livre e do espaço ocupado
    - Descritores e algoritmos de gestão de espaço
  - É necessário agregar a informação associada a um dado tema
    - Noção de ficheiro
  - É necessário organizar os ficheiros por tema ou afinidade
    - Noção de Directório
  - É necessário saber as características e a localização dos dados dos ficheiro
    - Descritor de ficheiro

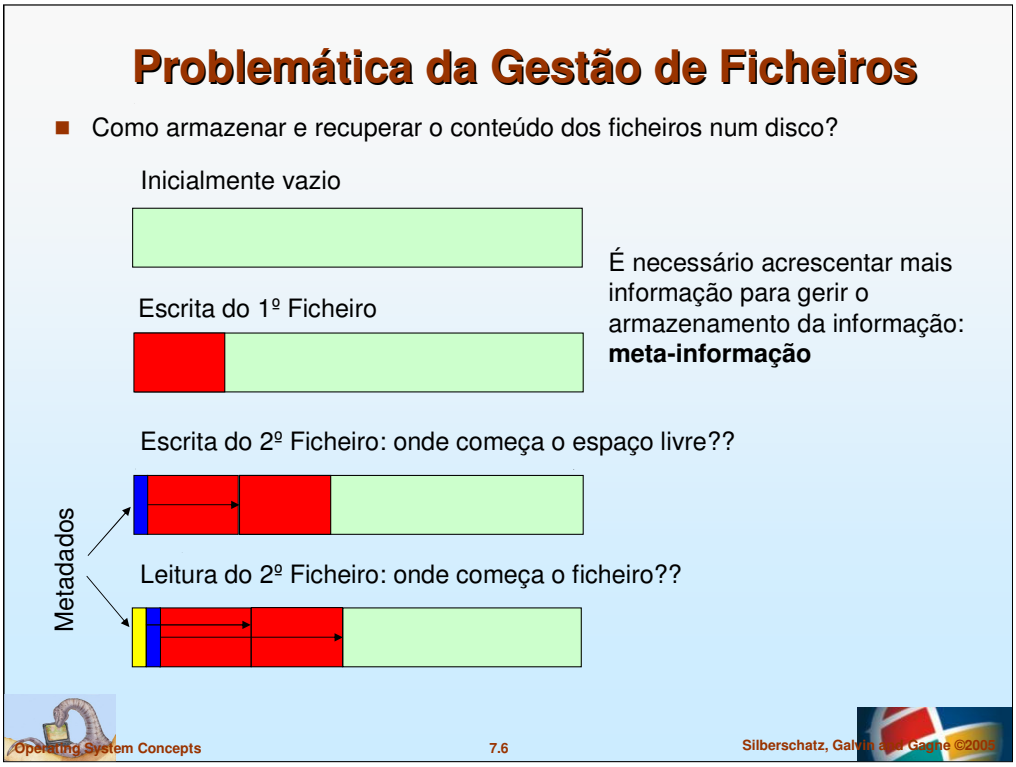


A API para manipulação de ficheiros é uma abstração com uma API conceptualmente simples. Mas essa simpicicade é conseguida à custa de várias camadas de software que escondem pormenores.

Níveis Funcionais

- Para poder implementar as funcionalidades de gestão de ficheiros, são necessários vários níveis funcionais:
  - Camada lógica que cria a noção de directórios e ficheiros
  - Camada funcional que realiza o mapeamento entre o ficheiro e a localização do seu conteúdo
  - Camada que realiza o acesso básico aos blocos de dados
  - Camada de Controle de I/O e de cache de blocos
  - Gestores de Periféricos

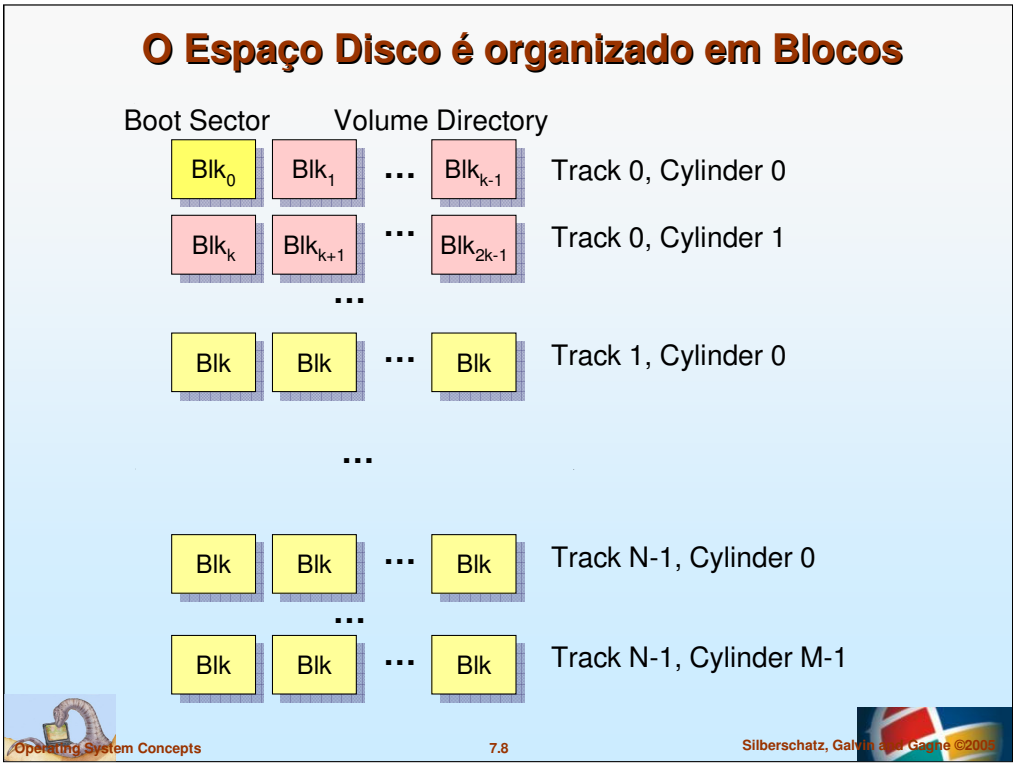




A meta-informação permite ao SO a gestão do espaço físico e outra informação adicional, suportada níveis mais altos (por exemplo datas de criação e modificação,etc)

Meta-informação

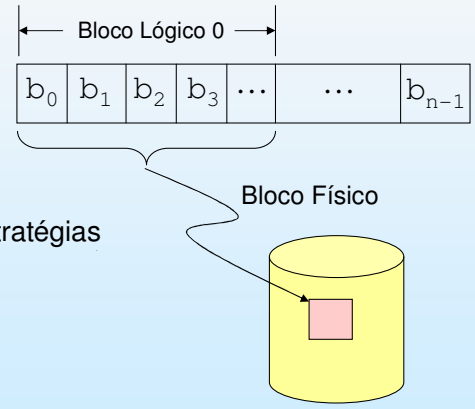
- Descritores de espaço
  - Lista dos espaços livres num disco
  - Tamanho, contiguidade, etc…
- Descritores de Ficheiros
  - Localização
    - Contígua: início, tamanho
    - Fragmentada: lista de blocos
  - Características
    - Tipo de Conteúdo
    - Protecções
    - Etc.
- Descritores do Meio de Armazenamento
  - Geometria
  - Partições
  - Etc.



A realidade física.

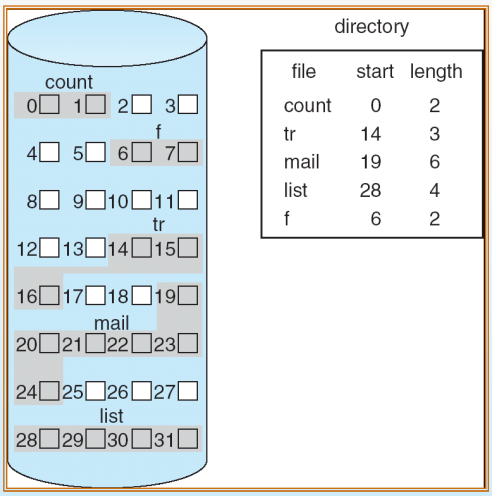
Gestão de Blocos

- É a funcionalidade que permite atribuir e seleccionar os blocos para armazenar o conteúdo dos ficheiros
- Para um tamanho de bloco fixo de b bytes
  - Um ficheiro de comprimento m necessita de k = | m / b | blocos
  - O byte de ordem b_i é armazenado no bloco | i / b |
- Podem ser utilizadas várias estratégias
  - Alocação contígua
  - Listas ligadas
  - Indexação



Alocação Contígua

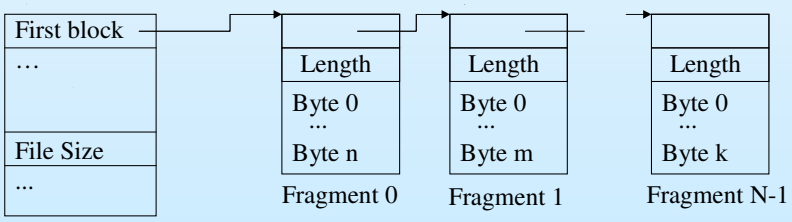
- Mapeamento da totalidade dos N blocos de um ficheiro em N blocos contínuos no disco
  - Solução simples
  - Permite acessos directos e rápidos
- Suporta muito mal a variação dinâmica do tamanho dos ficheiros
  - Necessidade de copiar o ficheiro todo para outra localização se não houver espaço contíguo disponível
  - Espaço livre não pode ser totalmente atribuído
  - Mesmos problemas que na alocação contínua de memória !
- Todavia, os SGFs mais recentes utilizam um esquema de alocação contínua modificado
  - Alocação contínua por extensão
    - Ex: Ext3 (Linux)



Tal como na gestão de memória, o modelo ideal (alocação continua) tem limitações. Temos que evoluir para mecanismos mais versáteis.

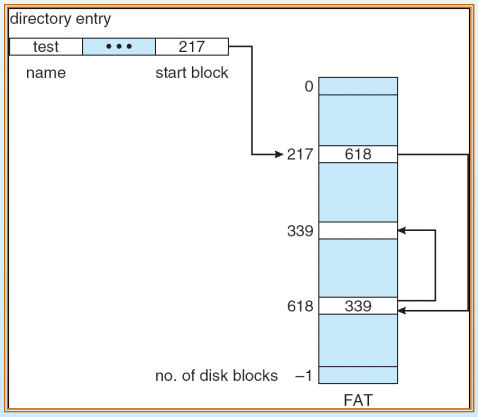
Listas Ligadas

- Um ficheiro é dividido em fragmentos
- Cada fragmento contém um cabeçalho de meta-informação
  - Número de bytes do fragmento
  - Ponteiro para o próximo fragmento
- Os ficheiros podem crescer e diminuir dinamicamente
  - Sem perca de espaço
  - Os fragmentos podem não ser contíguos
- O acesso aleatório pode ser lento
  - Necessita de aceder a todos os cabeçalhos até ao conteúdo pretendido

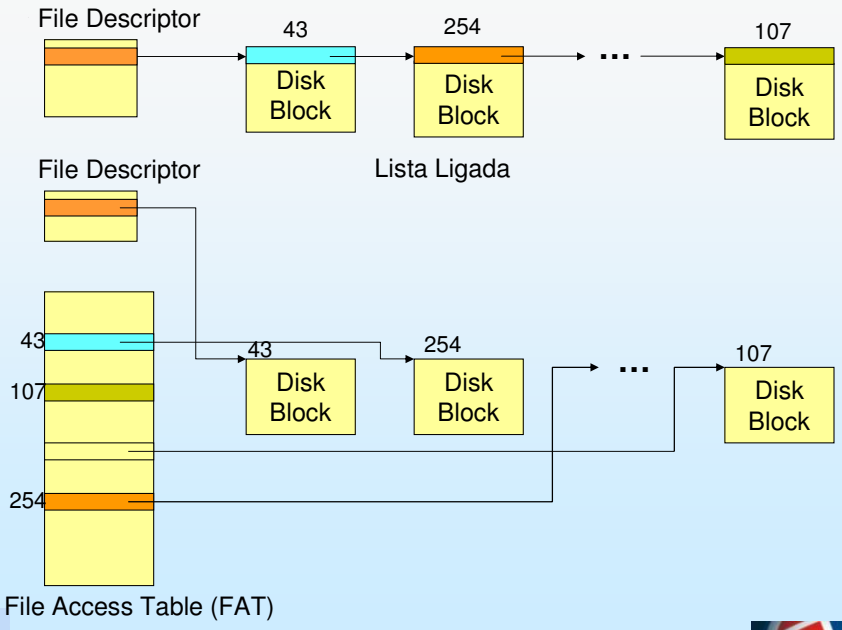


Exemplo: FAT do DOS

- O sistema DOS utiliza o conceito de File Access Table do DOS para implementar alocação em listas ligadas
- A tabela contém uma entrada por cada bloco do disco indexada pelo número de bloco
- Cada elemento da tabela contém o endereço do próximo bloco da lista alocada a um ficheiro
- O último bloco contém um caracter que indica o fim do ficheiro, ou alternadamente o file descritor pode conter o número do último bloco

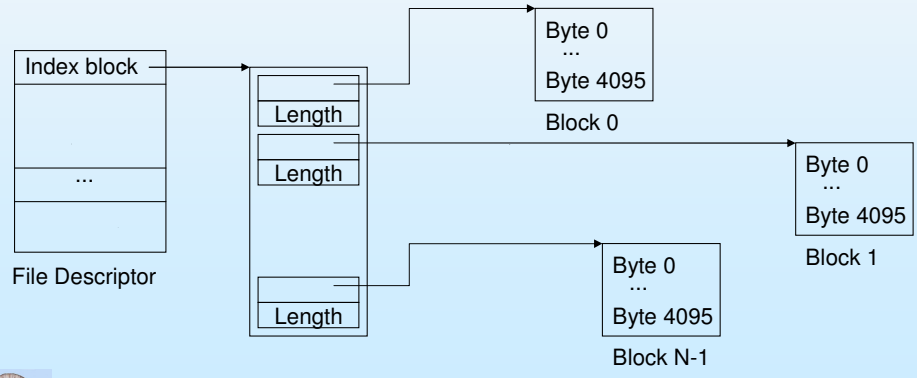


File Access Table



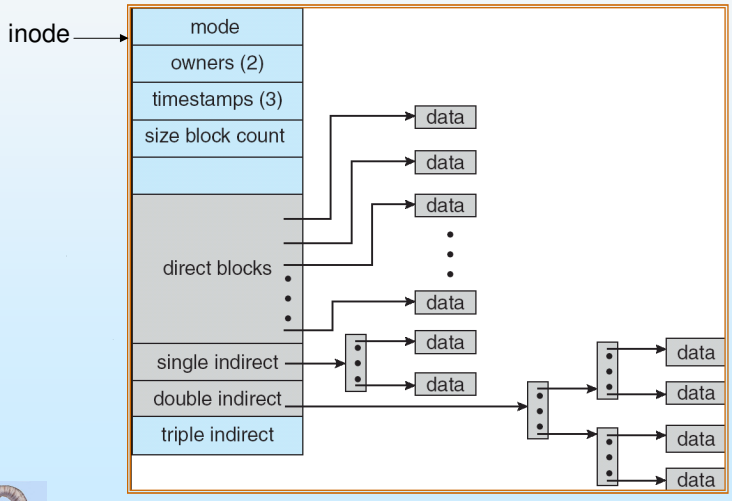
Alocação Indexada

- Agrupa todos os headers dos blocos atribuídos numa única lista indexada pelo numero de bloco, contida no primeiro bloco do ficheiro
- Simplifica o acesso aleatório ao conteúdo do ficheiro
- Para gerir ficheiros muito grandes, pode haver vários níveis de indexação (cf. tabelas de páginas hierárquicas)



Alocação no Unix File System (UFS)

- Utiliza o *file descriptor* (*inode*) para armazenar uma primeira lista de 12 blocos
- Utiliza dupla e tripla redirecção de endereços



Gestão do Espaço Livre

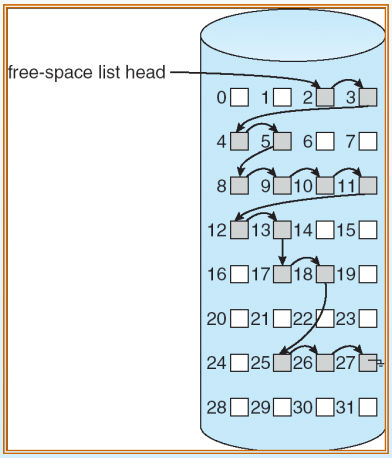
- A alocação dinâmica de espaço implica o armazenamento de informação sobre os blocos livres
  - Quando um ficheiro é criado ou é expandido torna-se necessário alocar blocos para armazenar o seu conteúdo
  - Quando um ficheiro é apagado, os blocos que continham o seu conteúdo têm de voltar a ser disponibilizados para alocação
- A alocação de blocos para armazenar o conteúdo de ficheiros é muito importante para o desempenho de um SGF
  - Escrita
    - Determina a rapidez com que se conseguem criar ou estender ficheiros
  - Leitura
    - Condiciona factores de localização do conteúdo do ficheiro tais como a fragmentação
- Existem inúmeros algoritmos para gerir o espaço livre em disco
  - Listas de blocos
  - Bitmaps
  - Extensões

Conceitos gerais. Não serão exigidos detaches.

Listas de Blocos

Os blocos livres são mantidos em listas ligadas

- **Simples**: cada bloco livre contém um ponteiro para o bloco livre seguinte
  - Solução pouco eficiente: para percorrer a lista é necessário aceder a todos os blocos livres
- **Agrupadas**: o primeiro bloco contém uma lista de outros N, sendo os primeiros N-1 livres, e o último um bloco contendo um ponteiro para mais N blocos
  - Solução utilizada nos primeiros sistemas Unix
  - Pouco eficiente relativamente à possibilidade de agregar zonas contíguas

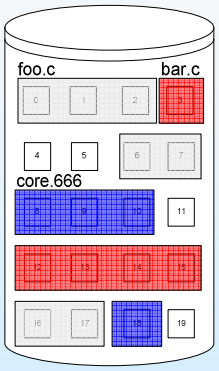


- **Agregadas**: cada elemento da lista contém o endereço e o tamanho do próximo agregado (*cluster*) de blocos livres e o seu tamanho em blocos
  - Mais eficaz pois permite alocação contígua ou por bloco consoante o tamanho do ficheiro

Termina aqui a matéria para o teste.

Alocação por Extensões

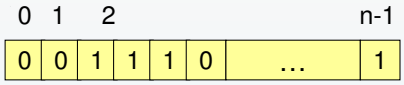
- O espaço livre é mantido numa lista de descritores contendo o número do bloco de início e o tamanho do agregado
  - Número reduzido de descritores
- Um ficheiro é constituído por uma lista de agregados de tamanho variável
- Permite operações de I/O rápidas quando limitadas a um agregado
- É fácil realizar a expansão ou redução de ficheiros
- A alocação pode ser feita utilizando técnicas simples do tipo first-fit
- Provoca fragmentação externa
  - Alguns SGFs permitem desfragmentação on-line
- Exemplo de SGF
  - Ext2 em Linux



Catalog

foo.c	(0,3)	(7,2)	(16,2)
bar.c	(3,1)	(12,4)	
core.666	(8,3)	(18,1)	

Vector de Bits



$$bit[i] = \begin{cases} 0 \Rightarrow \text{bloco } [i] \text{ livre} \\ 1 \Rightarrow \text{bloco } [i] \text{ ocupado} \end{cases}$$

- Os blocos do disco são representados através de um vector de bits
  - Permite compactar ao máximo a informação relativa aos blocos livres
- A obtenção dos blocos livres pode ser realizada através de instruções específicas para a manipulação de bits
  - O número de blocos livres consecutivos também pode ser facilmente obtido através da análise do padrão de bits
  - Para ser eficaz, o vector de bits deve estar residente em memória
- Para discos de grande capacidade o tamanho do vector pode tornar-se muito grande e difícil de manter
  - Incoerências entre a versão em memória e a do disco podem levar a perca de consistência de informação

Optimização de Desempenho

O desempenho do SGF é essencial para o do sistema como um todo

- Os acessos ao disco são muito frequentes e implícitos:
  - Execução de processos
  - Memória virtual
  - Alocação de espaço
- Inúmeras técnicas são utilizadas para optimizar a eficiência e o desempenho dos sistemas de ficheiros:
  - Descritores de ficheiros contendo endereços dos primeiros blocos dos ficheiros
  - Alocação de ficheiros em blocos contíguos ou fisicamente próximos para evitar múltiplos acessos em leitura
  - Utilização de *clusters* de blocos de tamanho variável para armazenar ficheiros com características diferentes
  - O *cache* sistema armazena os blocos mais frequentemente acedidos
  - Controladores de disco inteligente lêem pistas inteiras e armazenam os blocos em memória privada

Gestão da Coerência

- O maior problema que pode acontecer a um sistema de ficheiros é a perca de coerência da meta-informação que contém
- Pode suceder por inúmeras razões
  - Erro de programação (pouco frequente em sistemas estáveis)
  - Existência de diferenças entre dados no cache e em disco
  - Crash do sistema no decorrer de uma operação de I/O que modifica meta-informação
- Para evitar estes problemas existem várias soluções
  - Utilitários de teste e reparação da meta-informação
    - Fsck, chkdsk, etc
    - Permitem reparar inconsistências, mas não conseguem recuperar falhas graves do tipo perca de conteúdo de listas de blocos de ficheiro
  - Realizar as operações sobre meta-dados em escritas sincronas, para não manter meta-dados modificados em cache
- Utilização de Sistemas de Ficheiros com *Logs (Log File Systems)*
  - Escrevem em disco (num *log*) as operações que vão realizar antes de as realizar

Log File Systems

- Sistemas Gestão de Ficheiros que utilizam tecnologias de bases de dados para garantir a coerência da meta informação
  - Também designados por *Journaling* ou *Transactional File Systems*
    - JFS da IBM, NTFS do Windows, ou Ext3 do Linux
- Princípio de funcionamento:
  - O conjunto de operações de escrita de meta-dados associadas a uma modificação do conteúdo do disco é designado por *transacção*
  - O sistema guarda num *buffer* circular designado por *log* ou *jornal* uma lista sequencial de transacções, memorizando o índice da próxima a realizar
  - O *log* é armazenado numa zona específica do disco, de forma síncrona e ocupando blocos contíguos
  - As actualizações correspondentes a cada transacção são realizadas de forma assíncrona para os respectivos blocos do disco
  - Quando todas as operações de uma transacção tiverem sido efectivamente realizadas, esta é declarada válida (*commited*) e o índice é incrementado
  - Em caso de crash podem suceder vários casos:
    - As transacções que foram completadas antes do crash estão terminadas
    - As que não foram efectuadas ou não concluídas, são refeitas (*replayed*) a partir dos dados guardados no jornal
    - Se houver um registo no jornal não coerente, este é apagado (*undone*)
  - O jornal pode funcionar só com meta-dados ou com dados também, o que torna o sistema ainda mais seguro

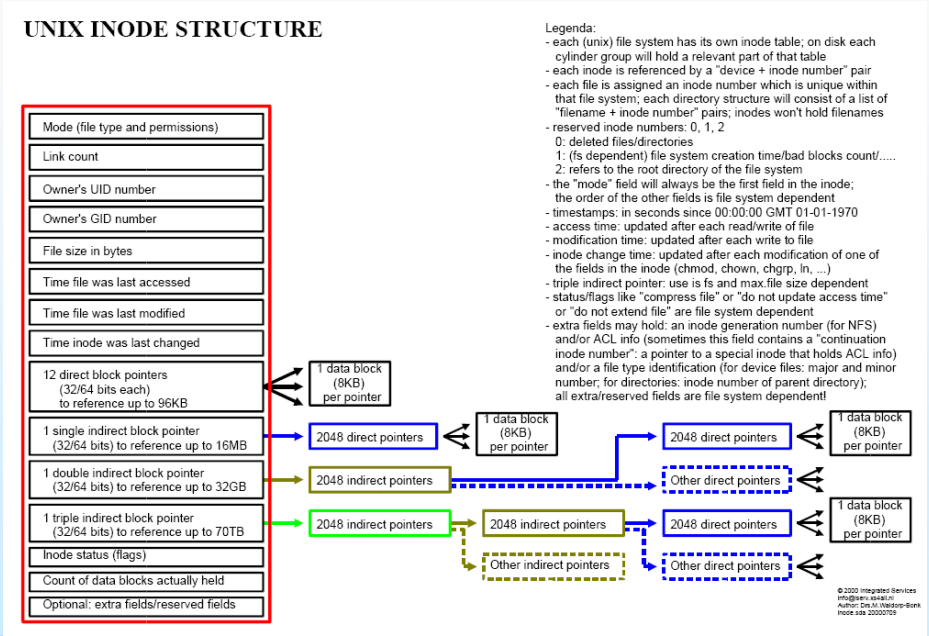
Interface do SGF

- Agrupa o conjunto de funcionalidades necessárias para gerir as principais representações do espaço de armazenamento
  - Ficheiros
  - Directorias
- Realizam o mapeamento entre a visão utilizador baseada numa sequência de bytes e a implementação possivelmente fragmentada do seu conteúdo em disco
- Ficheiro
  - Conteúdo não estruturado em SGFs como os de Unix e Windows, sendo a estrutura implementada pelas aplicações
  - Conjunto de registos (*records*) em sistemas de bases de dados ou mainframes, sendo a estrutura implementada pelo sistema
- Directório ou Catálogo
  - Permite o mapeamento entre os nomes (representação do utlizador) e o descritores internos (representação do sistema) dos ficheiros

Atributos de um Ficheiro

- Nome: a única informação armazenada em forma lisível
- Identificador: número único no sistema de ficheiros
- Tipo: reflecte o tipo de conteúdo, organização, etc..
- Localização
- Tamanho
- Datas de criação, modificação e acesso
- Identificação do proprietário
- A informação sobre ficheiros é guardada no respectivo catálogo, geralmente o nome e um ponteiro para o respectivo descritor
  - Também designado por **File Control Block**

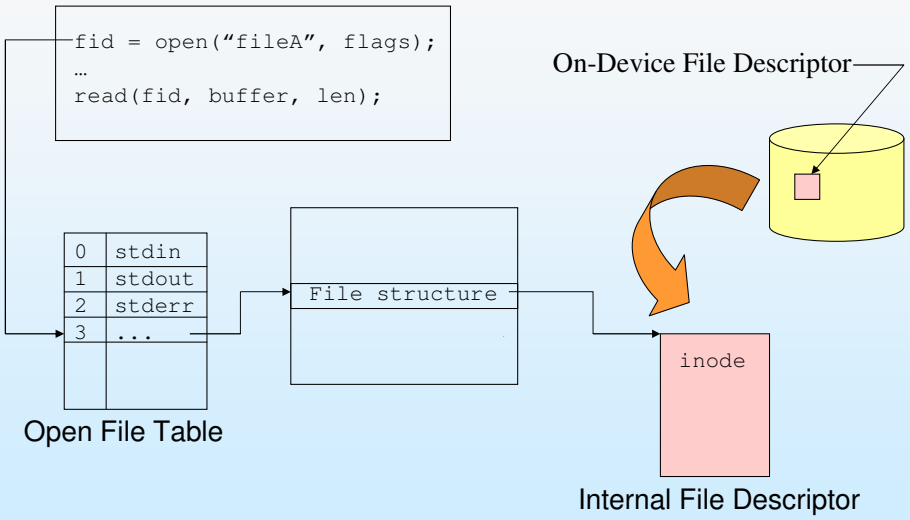
Estrutura do Inode Unix



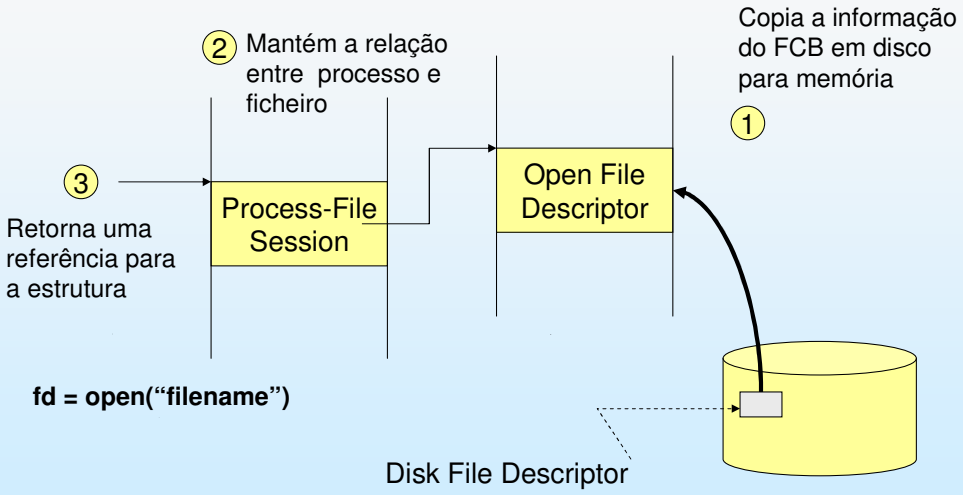
Operações Sobre Ficheiros

- **Open** - é a função que permite o mapeamento entre o nome de um ficheiro e o seu descritor interno (FCB)
  - Procura o nome do ficheiro num directório, acede ao seu FCB e traz o seu conteúdo para memória
  - Valida as permissões de acesso do utilizador
  - O sistema mantém uma tabela de ficheiros abertos, na qual são armazenados os respectivos FCBs
- **Close** - função inversa
  - Liberta o FCB em memória e realiza a actualização dos seus atributos
- **Create** - cria um novo ficheiro, alocando uma nova entrada no catálogo
- **Write** - transfere uma zona espaço do processo para dentro do ficheiro, realizando a alocação de espaço necessária à medida
- **Read** - função inversa
- **Seek** - permite posicionar o ponteiro de I/O em qualquer parte do conteúdo do ficheiro
- Etc…

Abertura de um ficheiro UNIX



Estruturas criadas pelo Open

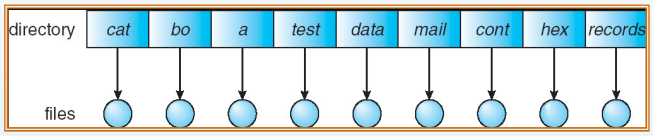


Estruturas de Directórios

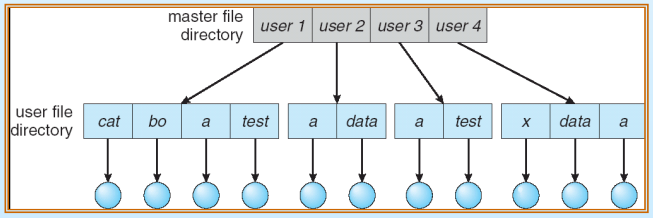
- Uma parte do volume contém os FCBs de todos os ficheiros existentes
  - Geralmente designado por **catálogo**
  - Unix: lista de i-nodes
- A criação de uma estrutura adicional é necessária para permitir uma organização do catálogo de forma mais eficiente e adequada para o utilizador
  - Eficiência
  - Facilidade de nomeação e agrupamento
- Utilização do conceito de directório
  - Ficheiros com conteúdos específicos, constituidos por uma lista de nomes e respectivos FCBs

Directórios com Níveis Finitos

- Nível único: problemas de nomeação e agrupamento

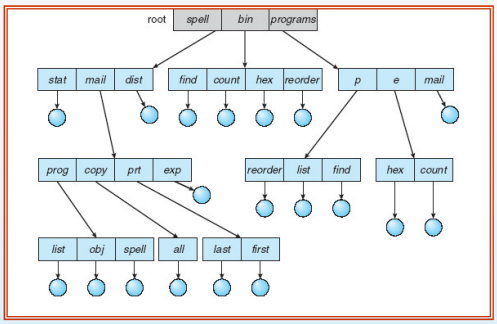


- Dois níveis:
- Noção de *pathname*
- Agrupamento rudimentar



Directórios em Árvore

- Forma mais adequada de organizar informação



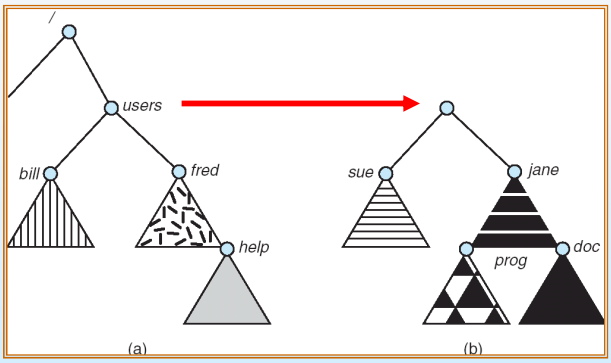
- Necessidade de um algoritmo de resolução de nomes de tipo recursivo
- Noção de directório corrente de cada processo
- Para resolver o nome /spell/mail/prog/list em Unix:
  - Começa pelo i-node da raiz (i-node nº 2 do catálogo)
  - Lê o conteúdo do directório e compara cada elemento da lista com o segundo elemento do nome
  - Em caso de igualdade, utiliza o número de i-node associado e recomeça com o terceiro elemento do nome

Operações sobre Directórios

- **Lookup**: procura de um ficheiro num directório, retorna um descriptor de ficheiro
- **Create**: Criação de um ficheiro
- **Delete**: remoção de um ficheiro
- **List**: listagem do conteúdo
- **Traverse**: atravessamento do directório na procura de um nó situado mais abaixo

Extensão de SGFs

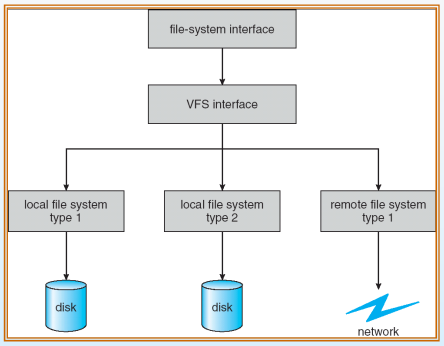
- Certos Sistemas de Ficheiros permitem a extensão do espaço de nomeação através da agregação de vários volumes
- Noção de montagem de volumes



- Inicialmente a) e b) são dois volumes distintos
- Depois da montagem formam um único volume lógico
- O espaço de nomeação é uniforme

Virtual File System

- A noção de Virtual File System fornece uma visão orientada aos objectos da implementação de sistemas de ficheiros



- É um nível de abstracção adicional introduzido entre a API de acesso aos ficheiros e a sua implementação em cada SGF
- As operações são realizadas de forma abstracta e instanciadas para cada SGF
  - Ex: a função read pode ser mapeada em ufs_read ou ext2fs_read conforme o tipo de SGF montado
- Permite realizar de forma transparente a distinção entre vários tipos de SGFs locais ou remotos

Fim