

Gestor de Sistemas de Ficheiros

Memória Persistente

Dispositivos de Memória Persistente

- **dispositivos lógicos:**
 - um dispositivo físico subdivide-se em segmentos contíguos chamados dispositivos lógicos, partições ou volumes
 - o sistema de ficheiros gere os acessos aos dispositivos lógicos
- **pode-se considerar um dispositivo lógico como um vector de blocos:**
 - os blocos têm uma dimensão múltipla da dos sectores (512 ou 1024 bytes) e, em máquinas de memória paginada, geralmente igual à das páginas
 - a informação dos ficheiros é guardada em grupos de blocos chamados segmentos (extents)
 - para diminuir a fragmentação é possível reaproveitar partes dos segmentos para outros ficheiros
- **superbloco:**
 - informação geral de descrição do sistema de ficheiros do dispositivo lógico

Discos Magnéticos

- **são compostos por:**
 - um conjunto de pratos sobrepostos
 - em cada face de cada prato a informação é escrita em pistas concêntricas, e cada pista é composta por um conjunto de sectores
 - o conjunto das pistas com o mesmo raio forma um cilindro
- **o tempo de leitura/escrita de um sector é composto por:**
 - tempo de posicionamento (seek time): tempo de deslocação das cabeças até ao cilindro desejado
 - tempo de latência: tempo de espera pelo sector (tmédio = tmeia rotação)
 - tempo de transferência: tempo que demora a transferir um sector entre o disco e a memória principal (trevolução / n. de sectores por pista)
- **tempo médio de acesso:**
 - soma dos tempos médios de posicionamento, latência e transferência

Optimização dos Acessos a Disco

- **minimização da frequência de acesso:**
 - cache de blocos em memória principal
- **minimização do tempo de posicionamento:**
 - é em geral muito superior ao tempo de latência
 - solução: ordenação dos pedidos segundo a ordem mais favorável:
 - **ordem de chegada:** simples, justo, não otimiza as operações mais demoradas (parar e mover cabeças)
 - **menor deslocamento:** menor tempo de posicionamento, maior desempenho quando os pedidos estão relacionados, pode ser injusto para os cilindros nas extremidades pois estes são preteridos
 - **elevador:** análogo ao anterior mas aplica-se apenas aos pedidos situados no sentido do deslocamento das cabeças, visita menos os cilindros na periferia (é dos mais usados)
 - **elevador circular:** cabeças lêem apenas num sentido, quando não há mais pedidos ou chegam à extremidade do disco as cabeças deslocam-se para o cilindro mais distante para o qual haja pedidos
- **minimização do tempo de latência:**
 - minimização do tempo de acesso a múltiplos sectores por pista - factor de entrelaçamento (interleave)
 - leitura de pistas inteiras para uma cache local ao controlador

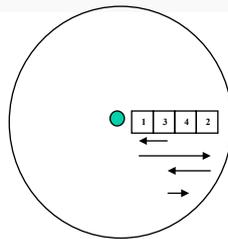
8/28/2003

Sistemas Operativos

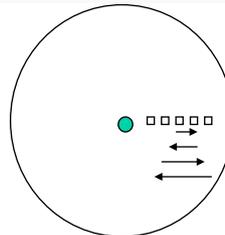
5

Ordenação dos Pedidos

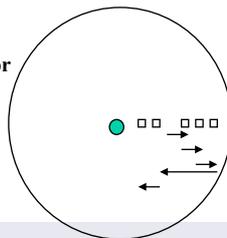
a) ordem de chegada



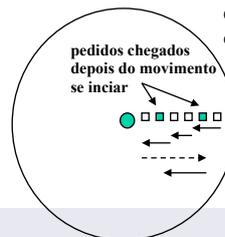
b) menor deslocamento



c) elevador



d) elevador circular



8/28/2003

Sistemas Operativos

6

Departamento de Engenharia Informática



INSTITUTO SUPERIOR TÉCNICO

Sistema de Ficheiros

8/28/2003 Sistemas Operativos 7

Departamento de Engenharia Informática



INSTITUTO SUPERIOR TÉCNICO

Sistema de Ficheiros

- O que é?
 - conjunto de ficheiros, directórios, descritores e estruturas de dados auxiliares geridos por um módulo do sistema operativo (**Sistema de Gestão de Ficheiros**)
 - permitem estruturar o armazenamento e a recuperação de dados persistentes em um ou mais dispositivos de memória secundária (discos ou bandas magnéticas)
- ficheiro
 - conjunto de dados persistentes, geralmente relacionados, identificado por um nome
 - é composto por:
 - nome: identifica o ficheiro perante o utilizador
 - descritor de ficheiro: estrutura de dados em memória secundária com informação sobre o ficheiro (dimensão, datas de criação, modificação e acesso, dono, autorizações de acesso)
 - informação: dados guardados em memória secundária

8/28/2003 Sistemas Operativos 8

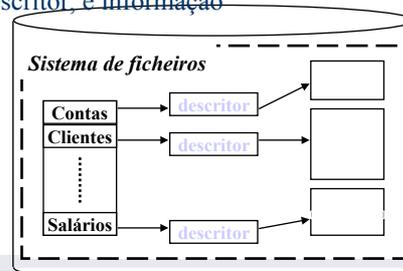
Sistema de Ficheiros (2)

- visão global:
 - um ficheiro é identificado por um caminho de acesso (*pathname*) podendo eventualmente existir diversos caminhos de acesso para o mesmo ficheiro
 - cada ficheiro tem associado um único *index node* (*inode*) que contém informação de localização e informação de gestão
 - o sistema encarrega-se de a partir do caminho de acesso, determinar o *inode* correspondente para todas as operações que envolvam ficheiros

Sistema de Ficheiros (3)

Directório:

- catálogo de nomes de ficheiros (ou directórios) que estabelece a associação entre os nomes e os seus descritores
 - Descritores são normalmente chamados *inodes*
- um directório pode conter os descritores ou apenas os seus identificadores
- um directório, tal como um ficheiro, também é composto por um nome, um descritor, e informação



Elementos a analisar do SGF

- **Modelo computacional:**
 - princípios gerais da interface programática com o sistema de ficheiros
- **Estrutura interna do sistema de ficheiros:**
 - organização da informação
- **Controlo dos dispositivos de memória secundária:**
 - controlo dos discos físicos, optimização da leitura/escrita, tamanho dos blocos em disco, etc.

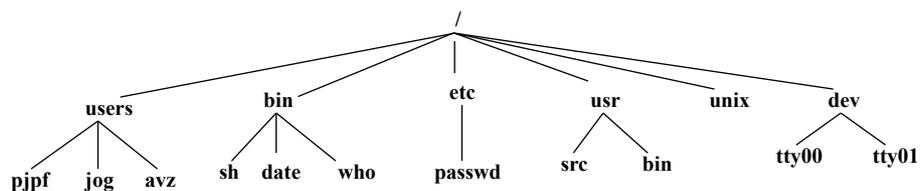
Modelo Computacional

- **Criação, abertura, fecho e eliminação de ficheiros:**
 - Criar (Nome, Protecção)
 - Fd = Abrir (Nome, Modo)
 - Fechar (Fd)
 - Eliminar (Nome)
- **Operações sobre ficheiros abertos:**
 - Ler (Fd, Tampão, NumRegistos)
 - Escrever (Fd, Tampão, NumRegistos)
 - Posicionar (Fd, PosiçãoRegisto)
- **Operações complexas sobre ficheiros:**
 - LerAtributos (Nome, Tampão)
 - AlterarAtributos (Nome, Tampão)
 - Mover (NomeOriginal, NovoNome)
- **Operações sobre directórios:**
 - ListaDir (Nome, Tampão)
 - MudaDir (Nome)
 - CriaDir (Nome, Protecção)

Organização dos Nomes dos Ficheiros

- Um nome por cada ficheiro num directório único a nível da máquina
- Um nome por cada ficheiro num directório único para cada utilizador
- Organização hierárquica (árvore):
 - solução proposta no **Multics**
 - os directórios contêm caminhos de acesso para nós descendentes a partir de um directório raiz
 - ficheiros e directorias vazias são nós terminais (folhas)
 - caminho de acesso (pathname): cadeia de caracteres que localiza um ficheiro ou directoria na árvore
 - nomes absolutos ou relativos:
 - absoluto: caminho de acesso desde a raiz
 - relativo: caminho de acesso a partir do directório corrente
 - directório corrente mantido para cada processo como parte do seu contexto

Organização Hierárquica



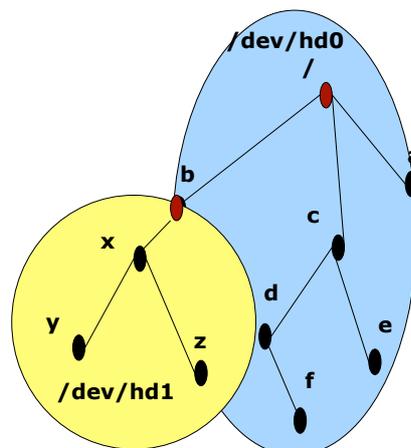
Sistema de Ficheiros em Árvore

Organização dos Nomes dos Ficheiros (2)

Uniformidade dos nomes:

- UNIX
 - ficheiros, directórios e dispositivos são referenciados usando a mesma sintaxe para o caminho de acesso
 - um dispositivo contendo um sistema de ficheiros pode ser montado num directório; a raiz desse sistema de ficheiros fica acessível através do caminho de acesso desse directório
- MS-DOS
 - espaço de nomes uniforme para ficheiros e directórios, tendo os dispositivos que ser referenciados de modo diferente (explícita ou implicitamente); existe o conceito de dispositivo corrente e de directório corrente por dispositivo
 - exemplo: A:\b\c\d
- VMS
 - não possui uniformidade de nomes
 - exemplo: a:[b.c]d

Uniformidade do Espaço de Nomes



`mount -t <filesystem> /dev/hd1 /b`

Organização dos Nomes dos Ficheiros (3)

- Extensões:
 - extensões convencionais (ex. UNIX)
 - o sistema de ficheiros não as reconhece, são mantidas pelos utilizadores e assumidas pelas ferramentas que manipulam ficheiros
 - extensões obrigatórias (ex. MS-DOS, VMS)
 - um nome de ficheiro é composto por nome.extensão (8+3 no MS-DOS), o separador “.” não faz parte do nome guardado pelo sistema de ficheiros; ambas as componentes podem ser nulas

Tipos de Ficheiros

O tipo de um ficheiro depende do seu conteúdo e forma de acesso:

- conteúdo: conjunto de registos (de dimensão fixa ou variável)
- Forma de acesso aos registos:
 - acesso sequencial:
 - para se ler o registo N é necessário ler os N-1 registos anteriores
 - para alterar um registo é preciso ler o ficheiro todo e escrevê-lo de novo com o registo alterado
 - acesso directo:
 - pode-se aceder directamente a um registo sem aceder aos anteriores
 - não se pode inserir um novo registo entre outros dois
 - acesso por chave:
 - os registos são identificados por chaves alfanuméricas reconhecidas pelo sistema de ficheiros (Indexed Sequential Access Method)

Protecção

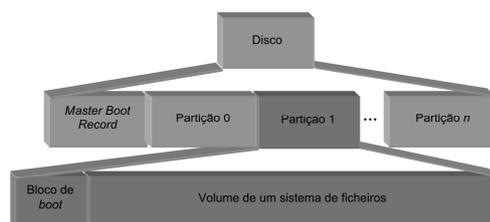
- **Estabelecimento da protecção quando o ficheiro é criado:**
 - A informação de protecção é guardada no descritor do ficheiro e está normalmente relacionada com o utilizador responsável pela sua criação, o qual é reconhecido como o seu dono
 - o dono de um ficheiro pode ser alterado durante o tempo de vida de um ficheiro
- **Verificação dos direitos de acesso de um processo a um ficheiro pelas funções sistema do sistema de ficheiros**
- **Especificação dos direitos de acesso a um ficheiro:**
 - listas de acesso para múltiplos utilizadores (Multics, opcional no VMS, Windows)
 - definição de grupos de utilizadores e de direitos de acesso a um ficheiro para o dono, para um grupo de utilizadores e para os demais utilizadores (UNIX)

Relação entre o Sistema de Ficheiros e as E/S

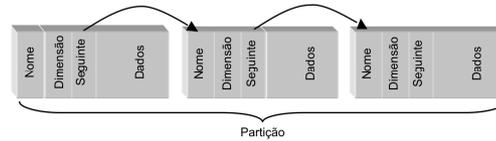
- **existem duas relações típicas:**
 - o sistema de ficheiros situa-se num nível hierárquico acima das Entradas/Saídas, sendo estas usadas para aceder aos periféricos de memória de massa (VMS)
 - as Entradas/Saídas estão ao mesmo nível do sistema de ficheiros, sendo os dispositivos virtuais vistos como quaisquer outros ficheiros (UNIX, Windows)
- **comparação:**
 - a segunda opção aumenta o número de funções sistema para manipular o sistema de ficheiros (devido à existência dos periféricos virtuais) mas facilita a redirecção das leituras e escritas para ficheiros, terminais, impressoras ou outros periféricos

Estrutura Interna do Sistema de Ficheiros

Entidades constituintes de um dispositivo de memória secundária

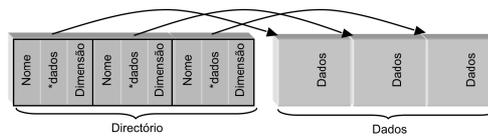


Organização dos ficheiros armazenados numa partição com lista ligada



- Problemas
 - Tempo para localizar ficheiros
 - Fragmentação do espaço
 - Dificuldade em crescimento dinâmico

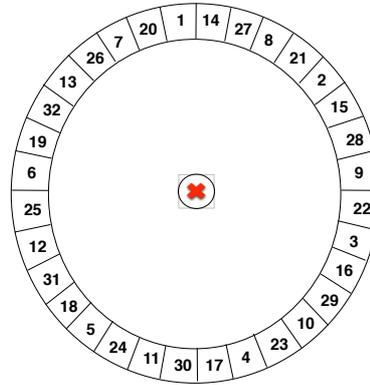
Solução em lista com um directório global único



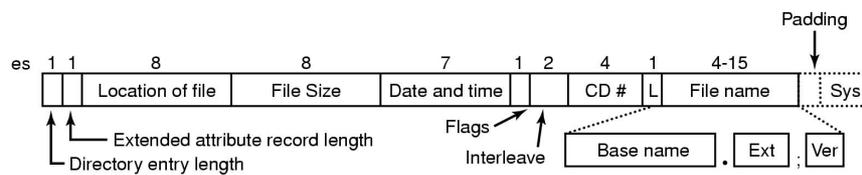
- Problemas
 - Fragmentação do espaço
 - Dificuldade em crescimento dinâmico
- Mas os CD-ROMs não têm estes problemas
 - Porque são apenas de leitura

Entrelaçamento - Interleave

- A ordenação dos sectores com um factor de entrelaçamento permite que se possa aceder a múltiplos sectores por pista.
- Consegue-se tratar um bloco e conseguir ler o sector seguinte porque a ordem lógica está desfasada da ordem física.

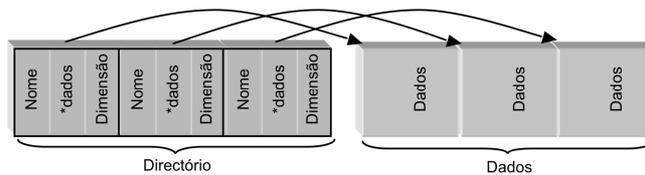


Sistema de Ficheiros para CD-ROM



- entrada de uma directoria em ISO 9660
- blocos de 2048 bytes (Modo-2: dados)
- não há gestão de espaço livre/ocupado
- ficheiros dispostos sequencialmente no CD-ROM

Solução em lista com um directório global único



- Problemas

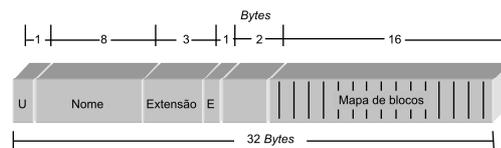
- Fragmentação do espaço
- Dificuldade em crescimento dinâmico

8/28/2003

Sistemas Operativos

27

Estrutura de uma entrada do directório do sistema de ficheiros do CP/M



- Problemas

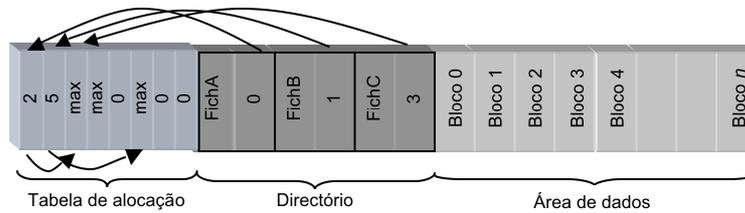
- Blocos de 1K
- Sistema de Ficheiros max 256K
- Ficheiro max 16K
- Aumentar implica aumentar todas as entradas de directório

8/28/2003

Sistemas Operativos

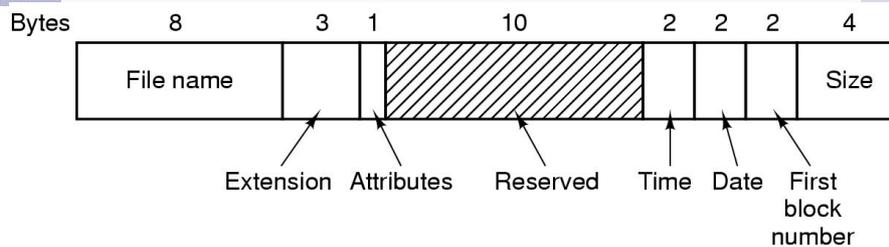
28

Estrutura de um sistema de ficheiros do tipo do sistema FAT



- A tabela de alocação é partilhada por todos
- Ficheiros grandes ocupam mais espaço na tabela de alocação
- Solução usada no MS-DOS a Windows 98

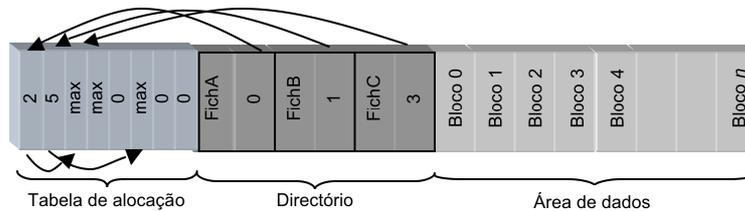
Sistema de Ficheiros MS-DOS (1)



entrada de 32 bytes de directoria em MS-DOS

- Usa uma FAT: 12, 16 ou 32 (28 bits)

Estrutura de um sistema de ficheiros do tipo do sistema FAT



- **Problemas**

- Grandes tabelas de alocação
- Disco 500G, Blocos de 4K e ponteiros de 32bit
 - $500G/4K \times 4 = 500M$
- Solução: aumentar a dimensão dos blocos, mas...

8/28/2003

Sistemas Operativos

31

Sistema de Ficheiros MS-DOS (2)

| Dimensão Bloco | FAT-12 | FAT-16 | FAT-32 (28) |
|----------------|--------|--------|-------------|
| 0,5 KB | 2 MB | | |
| 1 KB | 4 MB | | |
| 2 KB | 8 MB | 128 MB | |
| 4 KB | 16 MB | 256 MB | 1 TB |
| 8 KB | | 512 MB | 2 TB |
| 16 KB | | 1 GB | 4 TB (2 TB) |
| 32 KB | | 2 GB | 8 TB (2 TB) |

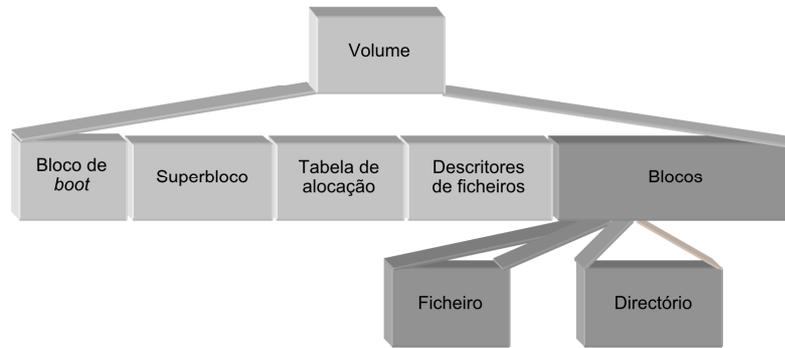
- Partição máxima para diferentes tamanhos de blocos
- As entradas a branco indicam combinações impossíveis (não permitidas)
- Apenas 28 dos 32 bits de cada entrada da FAT-32 são utilizados
- O sector de boot impõe duas outras limitações
 - Número de sectores por bloco: 8 bit
 - Número total de sectores no disco 32 bit

8/28/2003

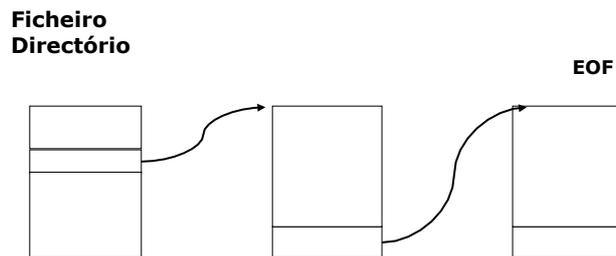
Sistemas Operativos

32

Partição com um sistema de ficheiros com descritores de ficheiros



Organização dos Blocos de Dados Lista Ligada

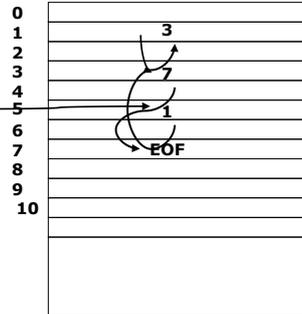
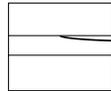


- **Organização Sequencial**
- **Acessos a disco directamente proporcionais aos blocos acedidos**

Organização dos Blocos de Dados Mapa da Memória Secundária

FAT – File Allocation Table

Ficheiro Directório



O Mapa é carregado em memória principal quando o disco é acedido

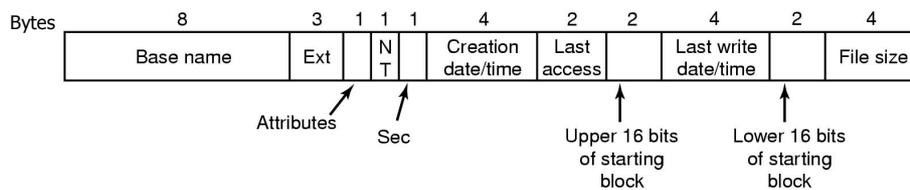
•Vantagens

- A selecção de um bloco pode ser feita em memória sem nenhum acesso a disco

•Desvantagem

- Tamanho da tabela directamente proporcional ao tamanho do disco

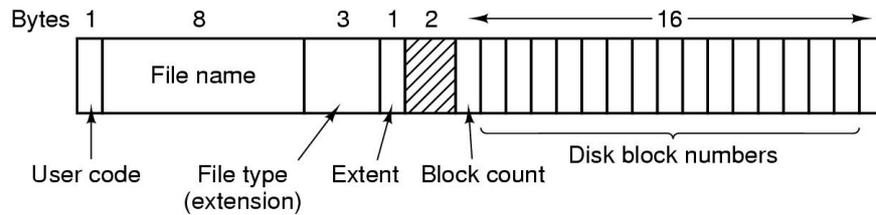
Sistema de Ficheiros Windows 98 (1)



entrada de uma directoria usada no Windows 98
(estendida do MS-DOS – utiliza os 10 bytes antes reservados)

- **Ficheiros com nomes grandes:** entrada **normal** com nome do tipo: SYSTEM~1.DOC, antecedida de entradas **adicionais** “**especiais**” que guardam o nome completo
- **Problema:** Como distingui-las e como manter compatibilidade com o DOS?

Sistema de Ficheiros CP/M

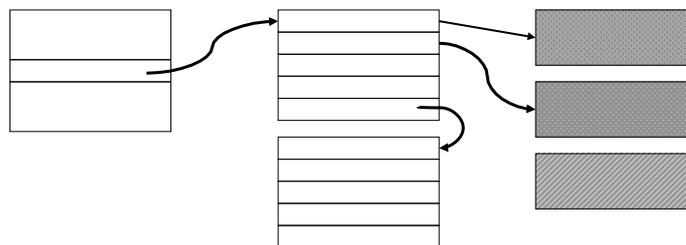


entrada de uma directoria em CP/M

- **Entrada de 32 bytes**
- **Blocos de 1kB**
- **Durante o arranque;**, é calculado um *bitmap* dos blocos livres a partir de todas as entradas e carregado em memória
- **Limite máximo de 16 blocos resolvido com vários extents p/ ficheiro**

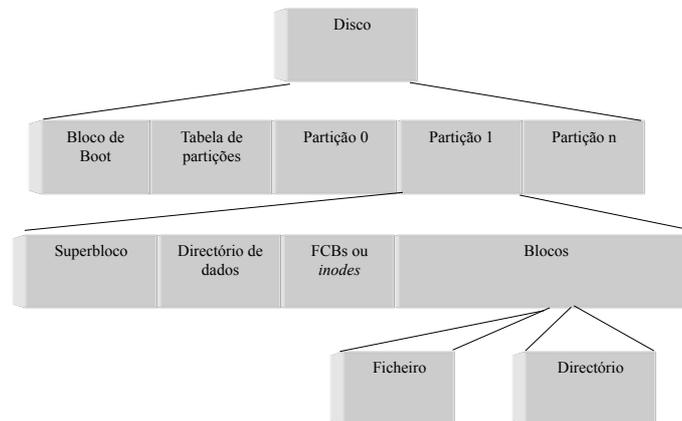
Organização dos Blocos de Dados Blocos de Índices

Ficheiro Directório



- Os blocos de índices são guardados em blocos de dados e só são acedidos quando necessário
- Adapta-se a qualquer dimensão do disco
- A estrutura de índices pode ser hierarquizada para otimizar o acesso directo em ficheiros de grande dimensão – **solução do Unix**

Entidades constituintes do Sistema de Ficheiros no dispositivo de memória secundária



8/28/2003

Sistemas Operativos

41

Qual o tamanho dos blocos ?

Impacto da fragmentação - Blocos grandes optimizam o I/O mas conduzem a desperdício de espaço por fragmentação interna

| Percent total waste | Percent data waste | Percent inode waste | Percent |
|---------------------|--------------------|---------------------|---|
| 0.0 | 0.0 | 0.0 | data only, no separation between files |
| 1.1 | 1.1 | 0.0 | data only, files start on 512-byte boundary |
| 7.4 | 1.1 | 6.3 | data + inodes, 512-byte block |
| 8.8 | 2.5 | 6.3 | data + inodes, 1024-byte block |
| 11.7 | 5.4 | 6.3 | data + inodes, 2048-byte block |
| 15.4 | 12.3 | 3.1 | data + inodes, 4096-byte block |
| 29.4 | 27.8 | 1.6 | data + inodes, 8192-byte block |
| 62.0 | 61.2 | 0.8 | data + inodes, 16384-byte block |

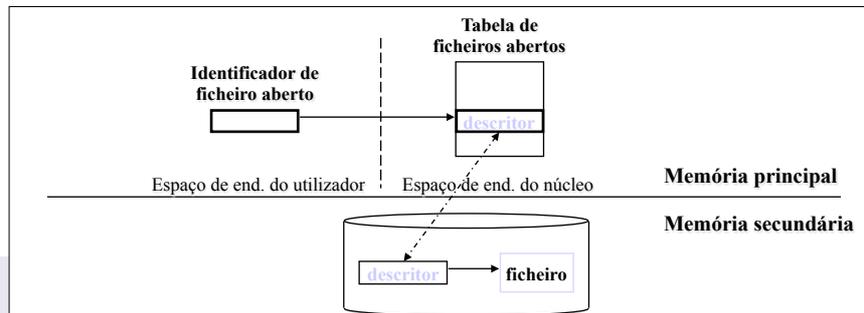
8/28/2003

Sistemas Operativos

42

Acesso a Ficheiros

- utilização de três etapas para aumentar o desempenho:
 - abertura do ficheiro dado o nome
 - é pesquisado o directório, copiado o descritor do ficheiro para memória (tabela de ficheiros abertos) e é devolvido ao utilizador um identificador de ficheiro aberto
 - verificar os direitos de acesso do processo ao ficheiro em causa
 - leitura ou escrita de informação dado o identificador de ficheiro aberto
 - este identificador permite chegar rapidamente ao seu descritor em memória
 - fecho do ficheiro



Unix – Sistema de Ficheiros

- O sistema de ficheiros Unix é uma das componentes iniciais do sistema e os principais conceitos mantêm-se válidos
 - i-nodes
 - Directórios
 - Ficheiros normais e especiais
- A evolução do sistema provocou várias optimizações
 - Virtual file system
 - Organização do disco
- No Linux Ext2 muitas das evoluções foram aplicadas

Evolução do Berkeley Fast Filesystem

- A organização do Unix tradicional tendia a espalhar a informação por muitos blocos do disco
 - Os i-nodes de ficheiros do mesmo directório ficavam em diferentes blocos
 - Os blocos eram separados em blocos físicos de 512 bytes em múltiplos cilindros
- O superbloco era crítico para a fiabilidade do sistema.

Evolução do Berkeley Fast Filesystem

- Solução
 - Organizar o disco em partições designadas **Cylinder Group**
 - Cada cylinder group tem um superbloco, área de i-nodes e blocos de dados
 - Os blocos de dados são de 4 Kbytes ou 8 Kbytes
 - Para evitar fragmentação são geridos fragmentos que podem ser de 1Kbyte.
 - O sistema tenta alocar os blocos de dados contíguos

Sistema de Ficheiros Linux

- Introdução
- Estruturas persistentes
- Estruturas voláteis

Introdução

- A primeira versão foi o EXT que segue as ideias do BFS
- Com a introdução do EXT2
 - Apareceu o VFS que permite vários sistemas de ficheiros (FAT, NTFS, EXT2, etc.)
- O SF vê os discos como vectores de blocos.
- O sistema de ficheiros só descreve a organização dos ficheiros pelos blocos, a escrita e leitura dos blocos é efectuada pelos gestores dos dispositivos.
- O SF gere a metadata dos ficheiros
 - Metadata persistente para todos os ficheiros
 - Metadata volátil para manter informação de ficheiros abertos

ESTRUTURAS PERSISTENTES

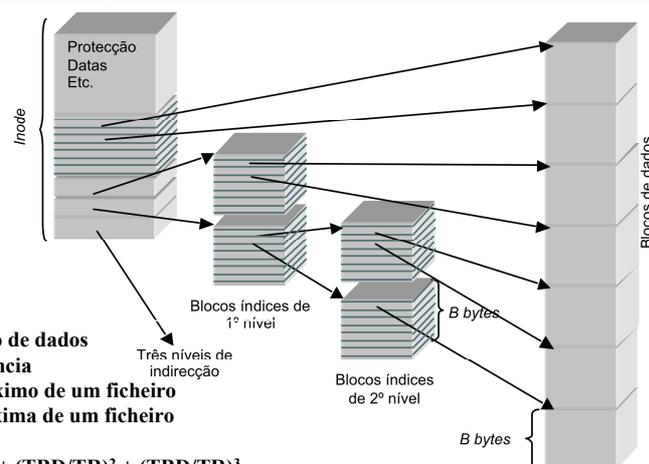
i-nodes

- Estruturas persistentes
- Contêm a identificação dos blocos no disco que fazem parte de um ficheiro.
- Existe um e só um i-node por ficheiro.
- Existem muitos tipos de i-node
 - Ext2, VFS, BSD
 - Todos têm estruturas diferentes mas têm o mesmo objectivo.

Visão interna de um i-node

| 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 <i>inode</i> . |
| 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 <i>hard links</i> . |
| i_blocks | Número de blocos ocupado pelo ficheiro. |
| i_flags | <i>Flags</i> várias do ficheiro. |
| i_block[15] | Vector de 15 unidades para blocos de dados. |
| | Outros campos ainda não utilizados. |

Utilização dos índices do i-nodes

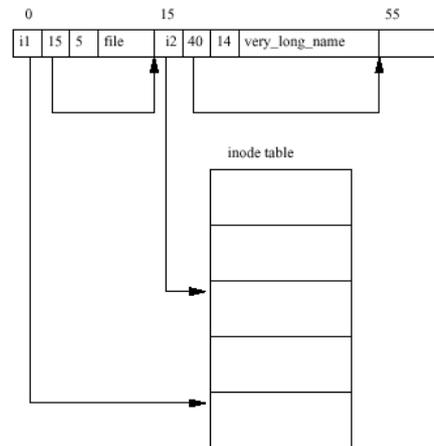


TBD – Tamanho bloco de dados
TR – Tamanho referência
Bmax – N° Blocos máximo de um ficheiro
Fmax – dimensão máxima de um ficheiro

$Bmax = 12 + TBD/TR + (TBD/TR)^2 + (TBD/TR)^3$
 com TBD 1024 e TR = 4, temos $Bmax \approx 16$ Megablocos
Fmax ≈ 16 Gigabytes

Directório no Ext2

- Um directório é um ficheiro como os restantes, que tem uma estrutura específica
- Um ficheiro do tipo directório contém um vector de entradas, em que cada entrada descreve um ficheiro desse directório.
- Cada entrada contém o nome do ficheiro a sua dimensão e o nº do i-node que o representa.
- Podem existir vários ficheiros com o mesmo i-node. São os *hard links*.
- Os *soft* ou *symbolic links* são ficheiros com o seu i-node com uma estrutura própria.
 - `ln -s ficheiroAlvo nomeDoLink`



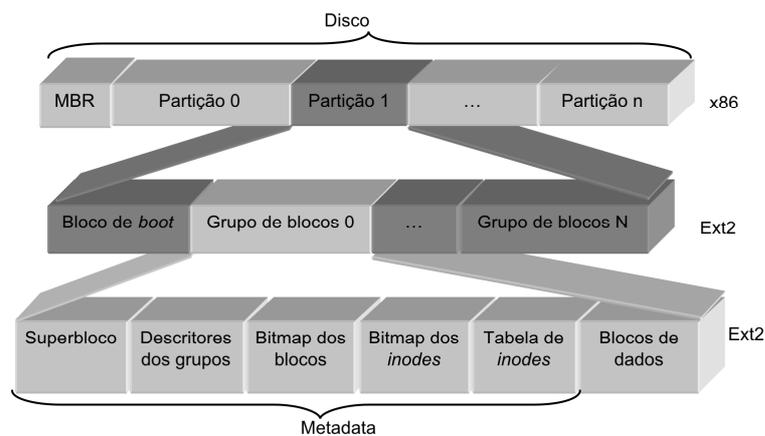
Ficheiro Directório

| | 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 |

Abertura de um ficheiro

- `/home/carlos.ribeiro/.cshrc`
 1. Procurar o i-node do directório raiz “/” no superbloco do dispositivo principal.
 2. Obter os blocos desse directório e encontrar a entrada “home”
 3. Ler o i-node do ficheiro tipo directório “home”.
 4. Ler os blocos de dados do “home”.
 5. Encontrar a entrada “carlos.ribeiro” e ler o seu i-node.
 6. Encontrar a entrada de “.cshrc” e ler o seu i-node e deste os blocos com a informação.

Restantes estruturas persistentes de um EXT2



Departamento de Engenharia Informática



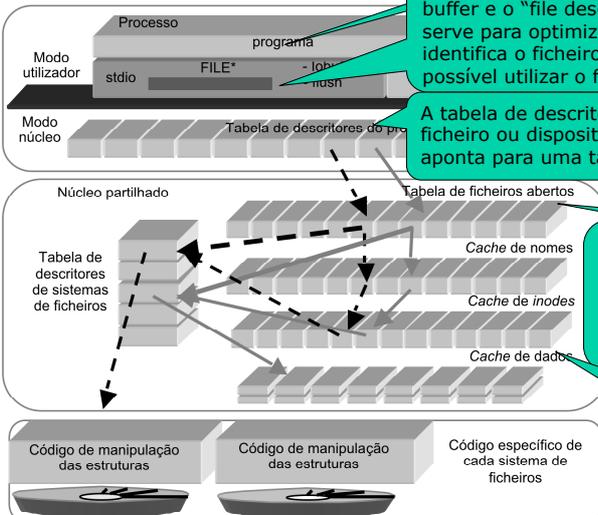
ESTRUTURAS VOLÁTEIS

8/28/2003 Sistemas Operativos 57

Departamento de Engenharia Informática



Estruturas voláteis



O programa contém um ponteiro para uma estrutura do tipo FILE.

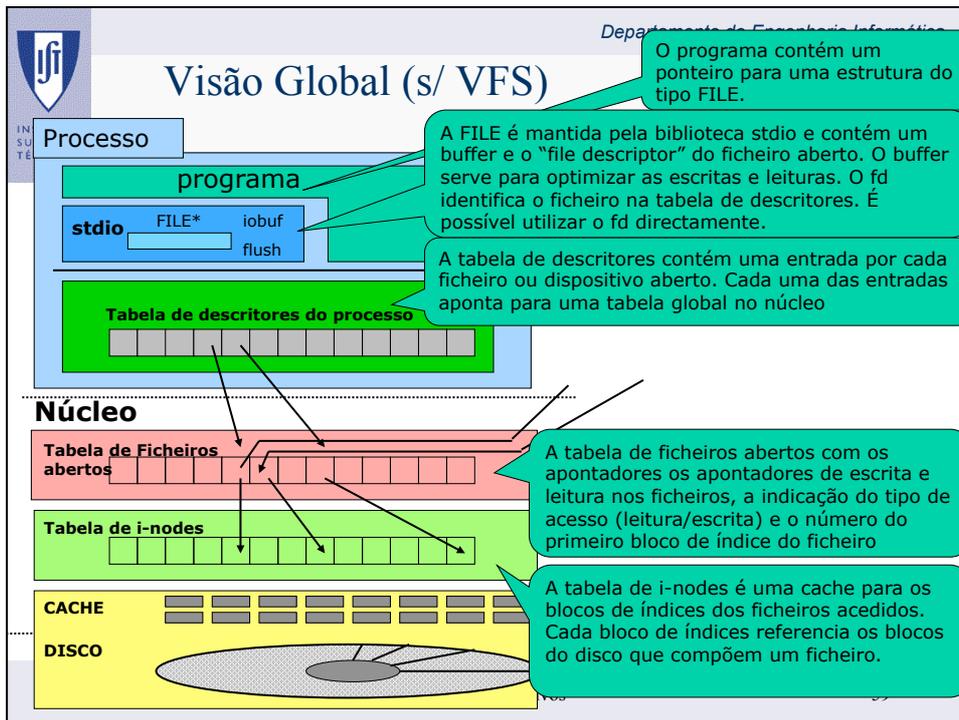
A FILE é mantida pela biblioteca stdio e contém um buffer e o "file descriptor" do ficheiro aberto. O buffer serve para otimizar as escritas e leituras. O fd identifica o ficheiro na tabela de descritores. É possível utilizar o fd directamente.

A tabela de descritores contém uma entrada por cada ficheiro ou dispositivo aberto. Cada uma das entradas aponta para uma tabela global no núcleo

A tabela de ficheiros abertos com os apontadores os apontadores de escrita e leitura nos ficheiros, a indicação do tipo de acesso (leitura/escrita) e um ptr p/ cache de nomes

A cache de i-nodes é uma cache para os blocos de índices dos ficheiros acedidos. Cada bloco de índices referencia os blocos do disco que compõem um ficheiro.

8/28/2003 Sistemas Operativos 58



Departamento de Engenharia Informática

Tabela de ficheiros abertos

- A tabela de ficheiros abertos contém uma entrada para cada ficheiro aberto
 - podem existir mais do que uma entrada para o mesmo ficheiro. Basta para tal este ser aberto por processos diferentes.
- No fork os ficheiros abertos pelo pai são partilhados pelo filho.
 - As entradas na tabela de ficheiros abertos também são partilhadas. Deste modo os cursores de leitura e escrita são partilhados o que permite direccionar a saída de pai e filho para o mesmo ficheiro sem que o último apague a saída do primeiro.

8/28/2003 Sistemas Operativos 60

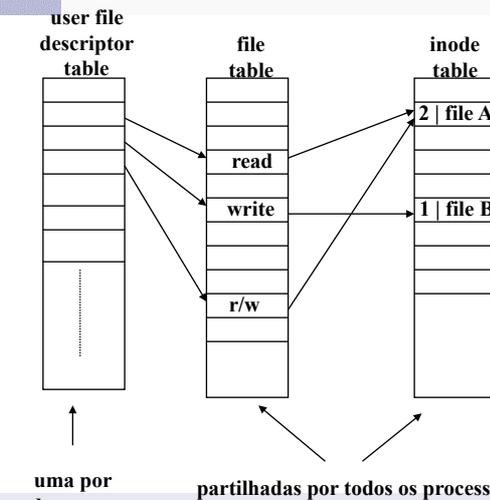
Entrada da tabela de ficheiros abertos

```

struct file {
    struct list_head    f_list;           // Ponteiro para o próximo elemento na lista
    struct dentry       *f_dentry;       // Ponteiro para o objecto dentry associado
    struct vfsmount     *f_vfsmnt;      // Ponteiro para o sistema de ficheiros
    struct file_operations *f_op;       // Ponteiro para a tabela de despacho
    atomic_t            f_count;        // Número de utilizações do ficheiro
    unsigned int        f_flags;        // Flags especificadas na abertura do ficheiro
    mode_t              f_mode;         // Modo de acesso
    int                 f_error;        // Código de erro para escrita em rede
    loff_t              f_pos;          // Posição actual de leitura ou escrita
    struct fown_struct  f_owner;        // Dados para notificação assíncrona
    unsigned int        f_uid, f_gid;   // Id do dono e do grupo
    struct file_ra_state f_ra;          // Dados para a leitura em avanço
    unsigned long       f_version;      // Versão, incrementada automaticamente
                                        // em cada uso
    void                *f_security;    // Estrutura de segurança genérica
                                        // (utilizada no SELinux)
    void                *private_data;  // Necessário para o tty
    struct list_head    f_ep_links;     // Lista de eventos para manipulação
                                        // assíncrona
    spinlock_t          f_ep_lock;      // Lock para protecção da lista de eventos
    struct address_space *f_mapping;    // Ficheiro mapeado em memória
};

```

Tabelas de Ficheiros



```

fd1 = open ("fileA", O_RDONLY);
fd2 = open ("fileB", O_WRONLY);
fd3 = open ("fileA", O_RDWR);

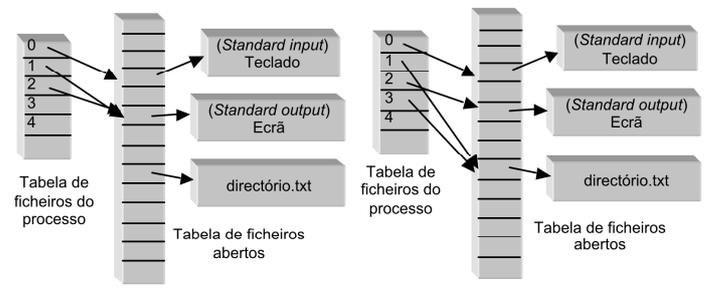
```

- *file table* contem:
 - Cursor que indica a posição actual de leitura/escrita
 - modo como o ficheiro foi aberto
- *processos pai e filho*:
 - partilham *file table*

Departamento de Engenharia Informática



Tabela de Ficheiros abertos



```

int fd= open("directorio.txt", "w"); // Abre o ficheiro de saída
close(1); // Liberta a entrada de indice 1 da tabela de ficheiros do processo
dup(fd); // Copia o conteúdo da entrada fd para a primeira entrada livre a l
execl("/bin/ls", "ls", 0); // Executa o comando ls neste processo

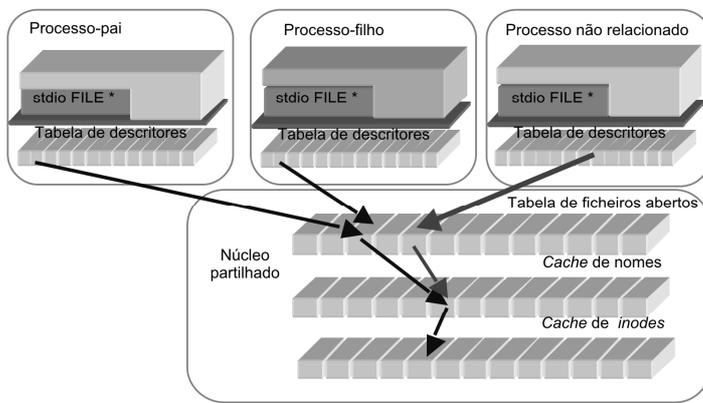
```

8/28/2003
Sistemas Operativos
63

Departamento de Engenharia Informática



Tabela Ficheiros Abertos: Fork



8/28/2003
Sistemas Operativos
64

Entrada da tabela de ficheiros abertos

```

struct file {
    struct list_head f_list;
    struct dentry *f_dentry;
    struct vfsmount *f_vfsmnt;
    struct file_operations *f_op;
    atomic_t f_count;
    unsigned int f_flags;
    mode_t f_mode;
    loff_t f_pos;
    unsigned long f_reada, f_ramax, f_raend, f_ralen, f_rawin;
    struct fown_struct f_owner;
    unsigned int f_uid, f_gid;
    int f_error;
    unsigned long f_version;
    void *private_data;
};

```

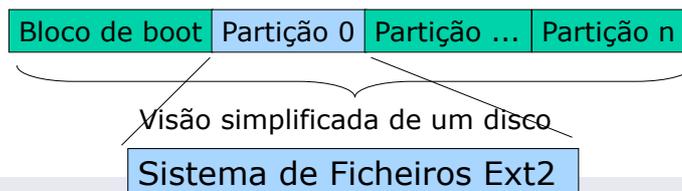
Campos relacionados com o VFS

Apontador para o ficheiro. Aponta para uma entrada de uma cache. Esta aponta para um i-node que possui os números dos blocos em disco do ficheiro

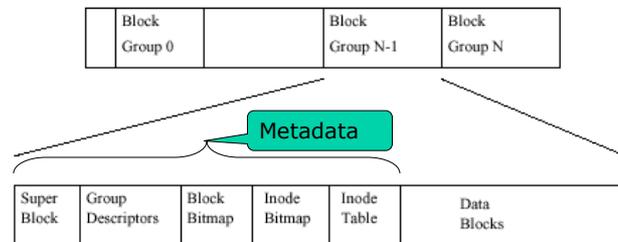
Campos que descrevem o estado do ficheiro. Nº de descritores que apontam para esta entrada, modo de abertura do ficheiro, ponteiros de leitura e escrita, etc.

Sistema de ficheiros nativo do Linux

- Denominado EXT2 (recentemente o EXT3).
- Descreve a organização física de uma partição de um disco.
- Um disco pode ter mais do que uma partição e em cada uma pode ter um sistema de ficheiros diferente.



Uma partição Ext2



- Cada partição está dividida em grupos de blocos.
- Cada grupo contém um conjunto de blocos com informação de ficheiros e um conjunto de informação sobre a estrutura desses ficheiros (metadata).
- Os blocos de um ficheiro podem estar distribuídos por vários grupos, mas devem preferencialmente estar no mesmo grupo.
- O objectivo dos grupos é manter a metadata “perto” dos dados que descreve, e assim reduzir os efeitos da corrupção de uma parte do disco.

Grupo de Blocos do Ext2

| | | | | | |
|-------------|-------------------|--------------|--------------|-------------|-------------|
| Super Block | Group Descriptors | Block Bitmap | Inode Bitmap | Inode Table | Data Blocks |
|-------------|-------------------|--------------|--------------|-------------|-------------|

- **Super Block (igual em todos os grupos)**
 - Contém a estrutura básica da partição (nº blocos, nº i-nodes, etc.)
- **Inode Table**
 - Contém uma tabela com os descritores dos ficheiros (i-nodes).
- **Inode Bitmap**
 - Contém um vector de bits que indica quais os i-nodes livres.
- **Block Bitmap**
 - Contém um vector de bits, que indica quais os blocos livres.
- **Group Descritores (igual em todos os grupos)**
 - Contém um descritor por cada grupo de blocos na partição.
 - Cada descritor contém, para o respectivo grupo, o nº de blocos e i-nodes no grupo, bem como onde está a tabela de i-nodes e as tabelas com de blocos e i-nodes livres.

Departamento de Engenharia Informática

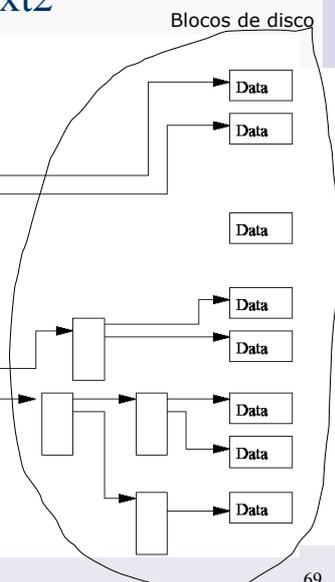


I-node do Ext2

ext2_inode

| |
|-----------------|
| Mode |
| Owner info |
| Size |
| Timestamps |
| Direct Blocks |
| Indirect blocks |

Blocos de disco



12 entradas directas

TBD – Tamanho bloco de dados
TR – Tamanho referência
Bmax – N° Blocos máximo de um ficheiro
Fmax – dimensão máxima de um ficheiro

$Bmax = 12 + TBD/TR + (TBD/TR)^2 + (TBD/TR)^3$
 com TBD 1024 e TR = 4, temos Bmax ≈ 16 Megablocos
 Fmax ≈ 16 Gigabytes

69

Departamento de Engenharia Informática



Ficheiro Directório

| | Inode | Dimensão do registo | Dimensão do nome | Tipo | Nome | | | |
|----|-------|---------------------|------------------|------|------|----|----|----|
| 0 | 54 | 12 | 1 | 2 | . | \0 | \0 | \0 |
| 12 | 79 | 12 | 2 | 2 | . | . | \0 | \0 |
| 24 | 23 | 16 | 6 | 1 | c | a | r | l |
| 40 | 256 | 16 | 7 | 1 | m | a | r | q |

8/28/2003

Sistemas Operativos

70

Abertura de um ficheiro

- `/home/carlos.ribeiro/.cshrc`
 1. Procurar o i-node do directório raiz “/” no superbloco do dispositivo principal.
 2. Obter os blocos desse directório e encontrar a entrada “home”
 3. Ler o i-node do ficheiro tipo directório “home”.
 4. Ler os blocos de dados do “home”.
 5. Encontrar a entrada “carlos.ribeiro” e ler o seu i-node.
 6. Encontrar a entrada de “.cshrc” e ler o seu i-node e deste os blocos com a informação.

Virtual File System

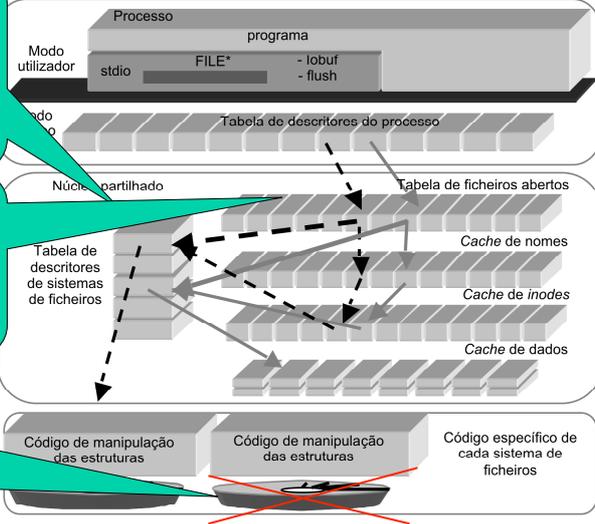
- Permite aceder a vários sistemas de ficheiros diferentes em simultâneo (EXT2, NTFS, FAT, NFS ...)
- Uma única hierarquia de ficheiros composta pelos vários sistemas de ficheiros.
- Facilita a construção de sistemas de ficheiros distribuídos.
- Permite a construção de sistemas de ficheiros virtuais tais como o `/proc`

Estruturas voláteis do VFS

Cada entrada:
 - é designada por VFS superblock.
 - contém informações sobre cada sistema de ficheiros
 - contém um vector de ponteiros para funções que sabem ler o sistema de ficheiros.

Cada entrada da Tabela de ficheiros abertos e cada entrada da tabela de inodes contém um apontador para o vector de funções que sabe manipular o sistema de ficheiros onde este se encontra.

Alguns sistemas de ficheiros não têm suporte físico. A informação é mantida noutros locais e.g. Núcleo, rede

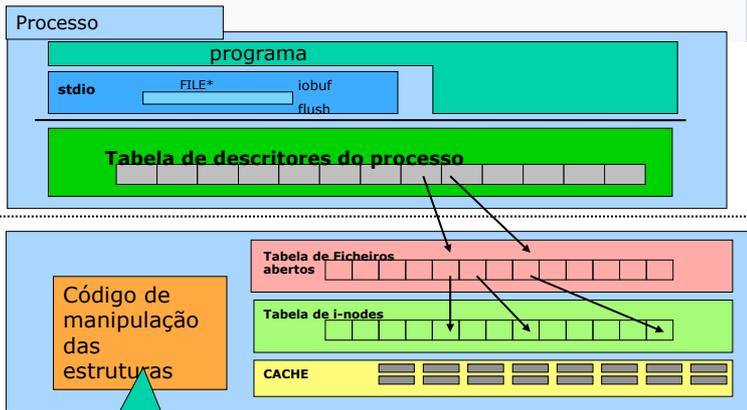


8/28/2003

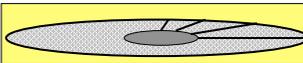
Sistemas Operativos

73

Um único Sistema de Ficheiros

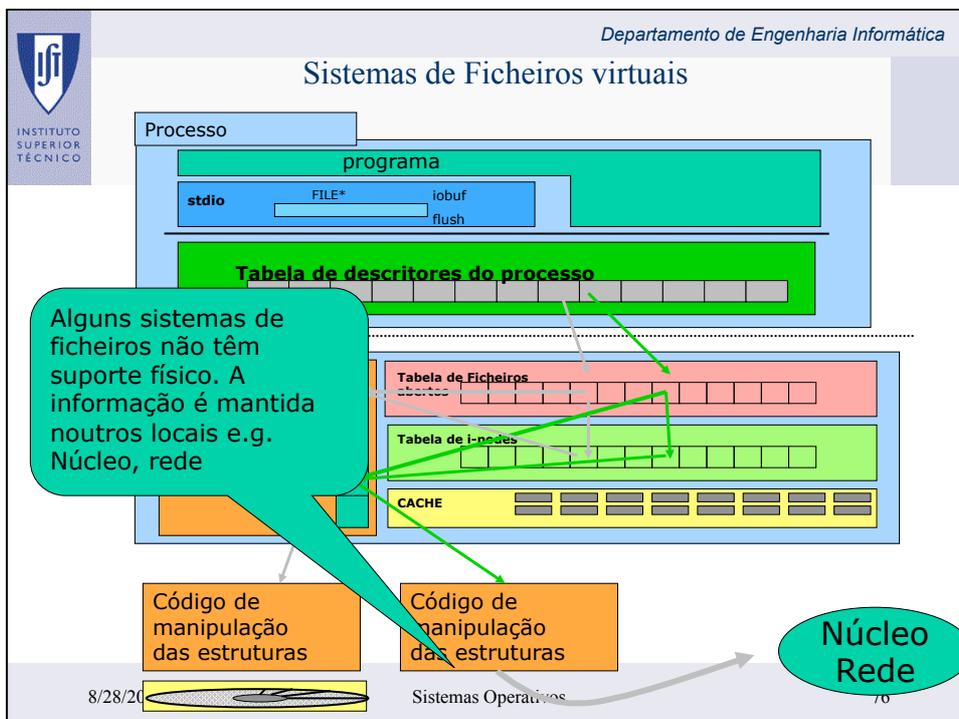
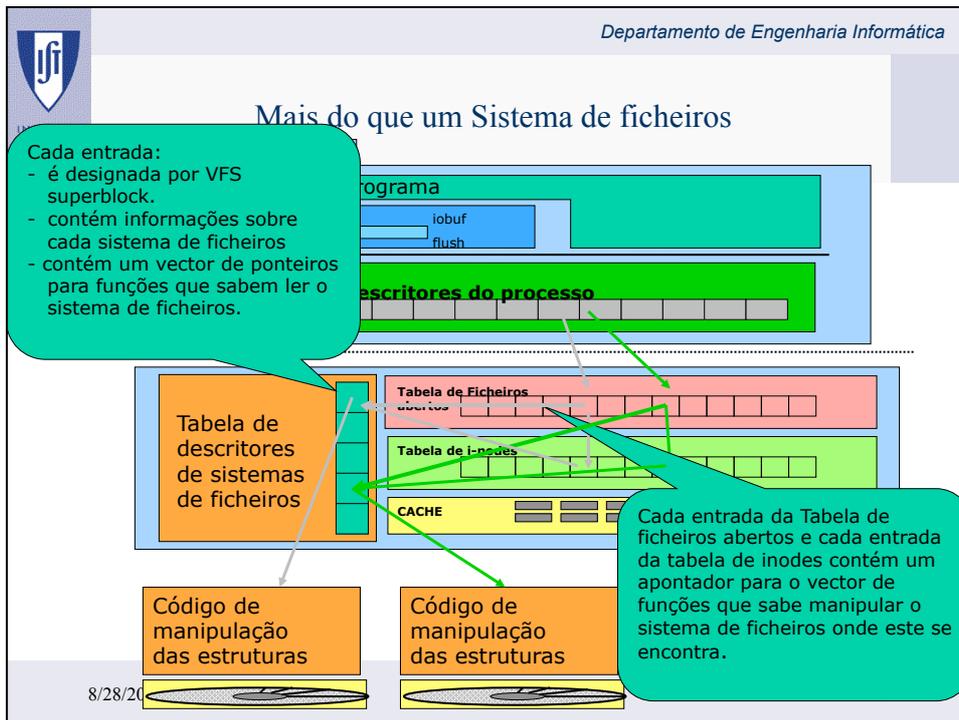


- Leitura de blocos do disco para a cache.
- Leitura de i-nodes do disco para a tabela de i-nodes
- Leitura de directórios



Sistemas Operativos

74



Caches

- Existem três tipos de caches no sistema de ficheiros do linux:
 - Cache de blocos de disco.
 - Cache de i-nodes.
 - Cache de directórios.

Cache de i-nodes

- Mantêm os i-nodes dos ficheiros abertos
- Só os i-nodes dos ficheiros abertos podem ser manipulados
- No VFS chamam-se v-nodes porque podem conter estruturas equivalentes a i-nodes de outros sistemas de ficheiros

Cache de Nomes

```

struct dentry {      /* Directory Entry
    atomic_t d_count;
    unsigned int d_flags;
    struct inode * d_inode;      /* Where the name belongs to - NULL is negative */
    struct dentry * d_parent;    /* parent directory */
    struct list_head d_hash;     /* lookup hash list */
    struct list_head d_lru;      /* d_count = 0 LRU list */
    struct list_head d_child;    /* child of parent list */
    struct list_head d_subdirs;  /* our children */
    struct list_head d_alias;    /* inode alias list */
    int d_mounted;
    struct qstr d_name;
    unsigned long d_time;        /* used by d_revalidate */
    struct dentry_operations *d_op;
    struct super_block * d_sb;   /* The root of the dentry tree */
    unsigned long d_vfs_flags;
    void * d_fsdata;            /* fs-specific data */
    unsigned char d_iname[DNAME_INLINE_LEN]; /* small names */
};

```

8/28/2003

Sistemas Operativos

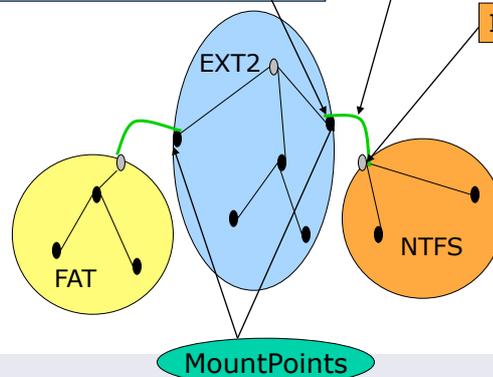
79

Organização global

```
mount -t ntfs -o rw /dev/hd1 /ficheirosWindows
```

dentry do directório /ficheirosWindows
marcado como mountpoint

Inode raiz do NTFS



8/28/2003

Sistemas Operativos

80

Cache de Blocos

- o núcleo poderia ler e escrever directamente para o disco em todos os acesso a ficheiros:
 - implicaria elevados tempos de resposta do sistema devido aos tempos de acesso ao disco
- para melhorar o desempenho:
 - minimizar os acessos ao disco através de uma cache que contém os blocos que foram recentemente acedidos
 - as rotinas de leitura e escrita analisam os blocos na cache antes de acederem ao disco
- dois níveis de cache:
 - biblioteca de I/O que adapta as operações de leitura/escrita ao tamanho dos blocos em disco
 - zona de memória entre os processos e os gestores dos discos

Cache de Blocos (2)

- cache para escrita/leitura em disco:
 - permitir manter em memória blocos de dados que possam ser reutilizados pelos processos
 - é constituída por blocos de memória em número que é um parâmetro de geração dos sistema
 - os blocos da cache têm dimensão igual à dos blocos em disco
 - os blocos na cache contém informação resultante de leituras/escrita anteriores do/para disco
- cada bloco da cache é representado por:
 - identificador do bloco
 - estado
 - apontador para os dados
 - apontadores que permitem a sua inclusão em listas duplamente ligadas

Departamento de Engenharia Informática



ESTRUTURA DOS BUFFERS

| |
|------------|
| device num |
| block num |
| status |
| |
| |
| |
| |
| |
| |
| |

prt to previous buf
on hash queue

prt to previous buf
on free list

prt to data area

prt to next buf
on hash queue

prt to next buf on
free list

device – identifica o sistema de ficheiros

bloco – identificador do bloco no sistema de ficheiros

8/28/2003
Sistemas Operativos
83

Departamento de Engenharia Informática



Cache de blocos

Hashed pelo device e pelo nº do bloco

Tabela de Hash

Listas com os blocos utilizados por ordem de utilização e pelo estado do bloco.

Listas LRU

Dirty Locked Clean

Listas de blocos livres organizadas por dimensão do bloco. Note-se que a cache é para ser utilizada por vários discos com diferentes dimensões de bloco

Listas de blocos livres

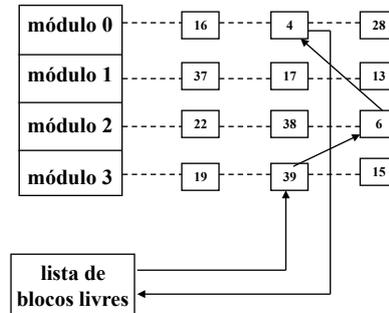
0.5K 1K 2K 4K 8K

Estrutura com o endereço e dimensão do buffer com o bloco de disco

8/2003
Sistem
84

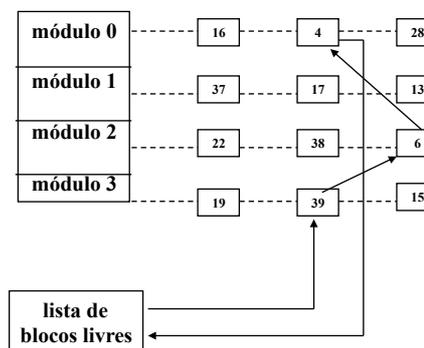
Cache (cont.)

- para cada radical da função de *hash*, os blocos são colocados numa lista duplamente ligada
- lista de blocos livres é gerida em LRU.
- um bloco não está livre quando está a ser usado para uma operação de leitura/escrita do/para o disco ou para o espaço de endereçamento do processo
- depois da operação antes referida, o bloco está livre
- quando o SO quer ler/escrever um bloco procura primeira na *cache* (função de *hash*)
- se o bloco procurado não está em memória, obtém o primeiro da lista de blocos livres



Cache de Blocos

- Na escrita não há acesso ao disco:
 - o bloco fica marcado como delayed write
 - o bloco fica também na lista de blocos livres
 - antes do bloco ser reutilizado, os dados são escritos para disco
 - periodicamente os blocos marcados como delayed write são escritos em disco
- Vantagem:
 - Apenas se efectua a escrita da última actualização do bloco. Se o bloco for imediatamente lido não há acesso ao disco
- Desvantagem;
 - A informação permanece volátil durante algum tempo e fica sujeita a perder-se se existir uma falha da máquina.



Departamento de Engenharia Informática

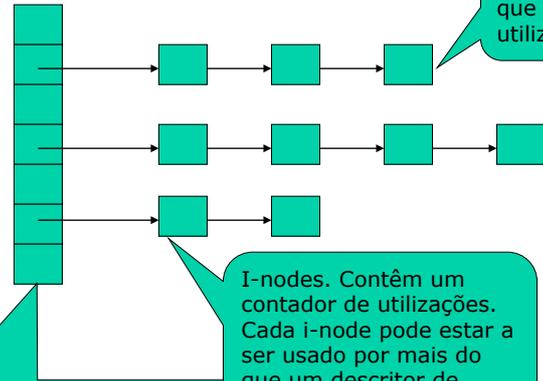


Cache de i-nodes

Tabela de inodes

Tabela de Hash

Hashed pelo número do dispositivo e pelo i-node



Os i-nodes candidatos à substituição são os que têm o contador de utilizações a zero.

I-nodes. Contêm um contador de utilizações. Cada i-node pode estar a ser usado por mais do que um descritor de ficheiro.

8/28/2003
Sistemas Operativos
87

Departamento de Engenharia Informática



Cache de directórios

Tabela de Hash

Hashed pelo device e pelo nome do directório

LRU – Least Recent Used com dois níveis.

1. Cada nova entrada é inserida no final da lista de nível um.
2. Quando 1 entrada é acedida segunda vez é colocada no final da lista de nível dois.
3. De cada vez que se coloca um elemento no final de cada uma das listas e não existem mais blocos para reservar é removido o elemento que está no topo dessa lista.

Cada entrada contém um componente do pathname, o device onde esse ficheiro está e o inode.

8/28/2003
Sistemas Operativos
88

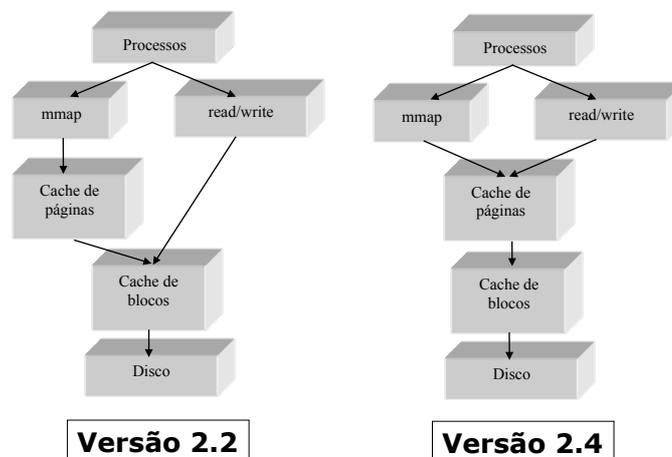
ACESSO A FICHEIROS MAPEADOS EM MEMÓRIA

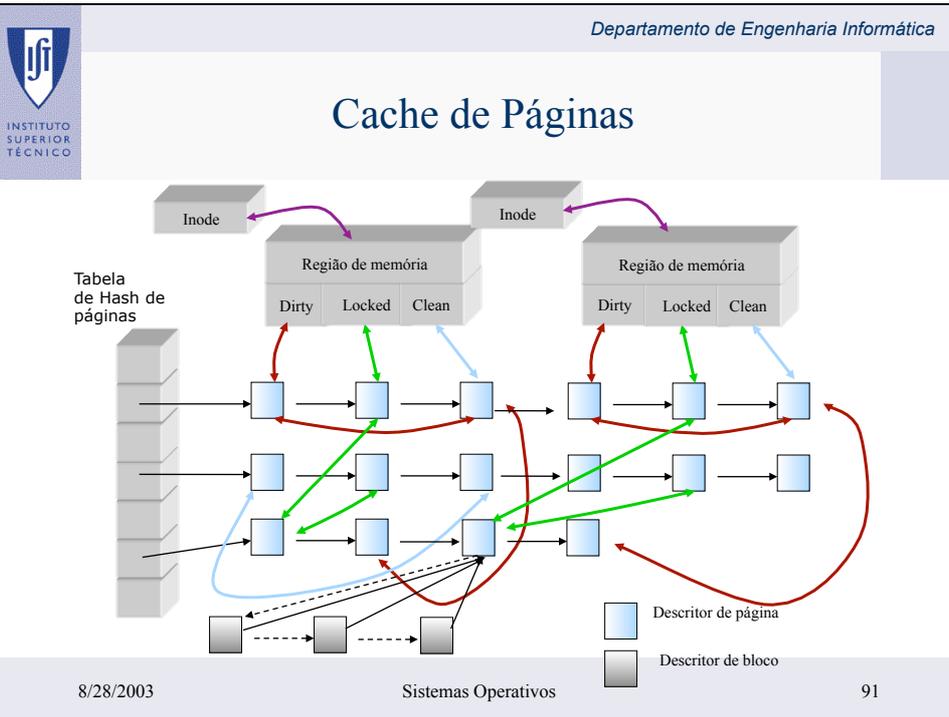
MapearFicheiro (Fd, posição, endereço, dimensão)

DesmapearFicheiro (endereço, dimensão)

- A primitiva MapearFicheiro permite aceder ao conteúdo de um ficheiro da mesma forma que se acede a uma qualquer outra estrutura em memória.
- O conteúdo do ficheiro indicado por Fd é:
 - mapeado a partir da posição indicada pelo parâmetro posição,
 - no endereço de memória a indicado pelo parâmetro endereço,
 - numa extensão indicada pelo parâmetro dimensão.
- A partir deste momento é possível aceder directamente ao ficheiro acedendo directamente às posições de memória indicadas.

Evolução da Cache em Linux – Suporte a ficheiros mapeados em memória





- Departamento de Engenharia Informática
- ## Reverse Mapping
- Versão 2.6 do Linux
 - Cada page-frame contém um rmap
 - Cada rmap aponta para as regiões de memória que contêm todas as páginas virtuais que partilham a mesma página física
 - Quando o page-stealer se executa pode libertar qualquer página seja ela pertencente a um ficheiro ou não
 - A gestão de memória é pois integrada
- 8/28/2003 Sistemas Operativos 92

Cache e consistência do sistema

- A política de cache agressiva praticada pelo Linux é particularmente útil para o desempenho global do sistema, mas é prejudicial para a garantia de persistência dos dados.
- Por exemplo, para efectuar a escrita de dados no final de um ficheiro é necessário:
 - localizar um bloco livre,
 - actualizar o *bitmap* de blocos livres situado no grupo de blocos do sistema de ficheiros
 - guardar o número do bloco no *inode* do ficheiro a que o bloco vai pertencer,
 - escrever o conteúdo do bloco
 - actualizar um conjunto de elementos do *inode*.
- Se durante uma destas operações existir uma falha do sistema, o ficheiro poderá ficar corrompido.

Cache e consistência do sistema

- O problema é resolvido marcando o sistema de ficheiros como sujo no início do sistema e só o considerar limpo se for correctamente desassociado, por uma operação explícita ou por uma operação de *shutdown*.
- Se no arranque se verificar que o sistema de ficheiros está sujo é executada uma operação de verificação (*fsck*) de todos os *inodes*, blocos ocupados e blocos livres existentes nessa partição.
- Esta operação sempre foi uma operação demorada, mas com a dimensão dos discos actuais tornou-se intolerável.

Journaling

- O objectivo da camada do núcleo *Journal Block Device* (JBD) é impedir que o sistema de ficheiros fique num estado inconsistente.
 - A JBD é actualmente utilizada apenas pelo sistema de ficheiros EXT3 (terceira versão do sistema de ficheiros *extend*).
- O JBD evita que uma operação de escrita seja parcialmente realizada, sendo efectuada de forma atómica.
- O JBD escreve os blocos a actualizar no *journal* e só depois de se ter assegurado que a informação já existe de forma persistente, é que a actualiza nos blocos do sistema de ficheiros. Se acontecer uma falha nesse período o sistema operativo aquando do reinício pode recuperar e refazer a escrita pois ela estava registada no *journal*.

Journaling

- A dupla escrita, no *journal* e no sistema de ficheiros, introduz uma penalização no desempenho do sistema pelo que muitos sistemas de *journal* só garantem a consistência dos metadados do sistema de ficheiros.
- O JBD permite configurar o tipo de *journaling* que se pretende efectuar, na associação do sistema de ficheiros a um directório (*mount*).
- Para além das duas opções já referidas o JBD fornece ainda uma terceira opção (por omissão) em que apenas os metadados são enviados para o *journal*, mas os blocos de dados relativos a um bloco de metadados são escritos primeiro que os metadados