Sistema de Ficheiros

Ficheiro

- Uma abstração que todos conhecem, desde a escola primária …

Conceito

- Coleção de dados persistentes, geralmente relacionados
- identificado por um nome
- Organizado em hierarquia de pastas

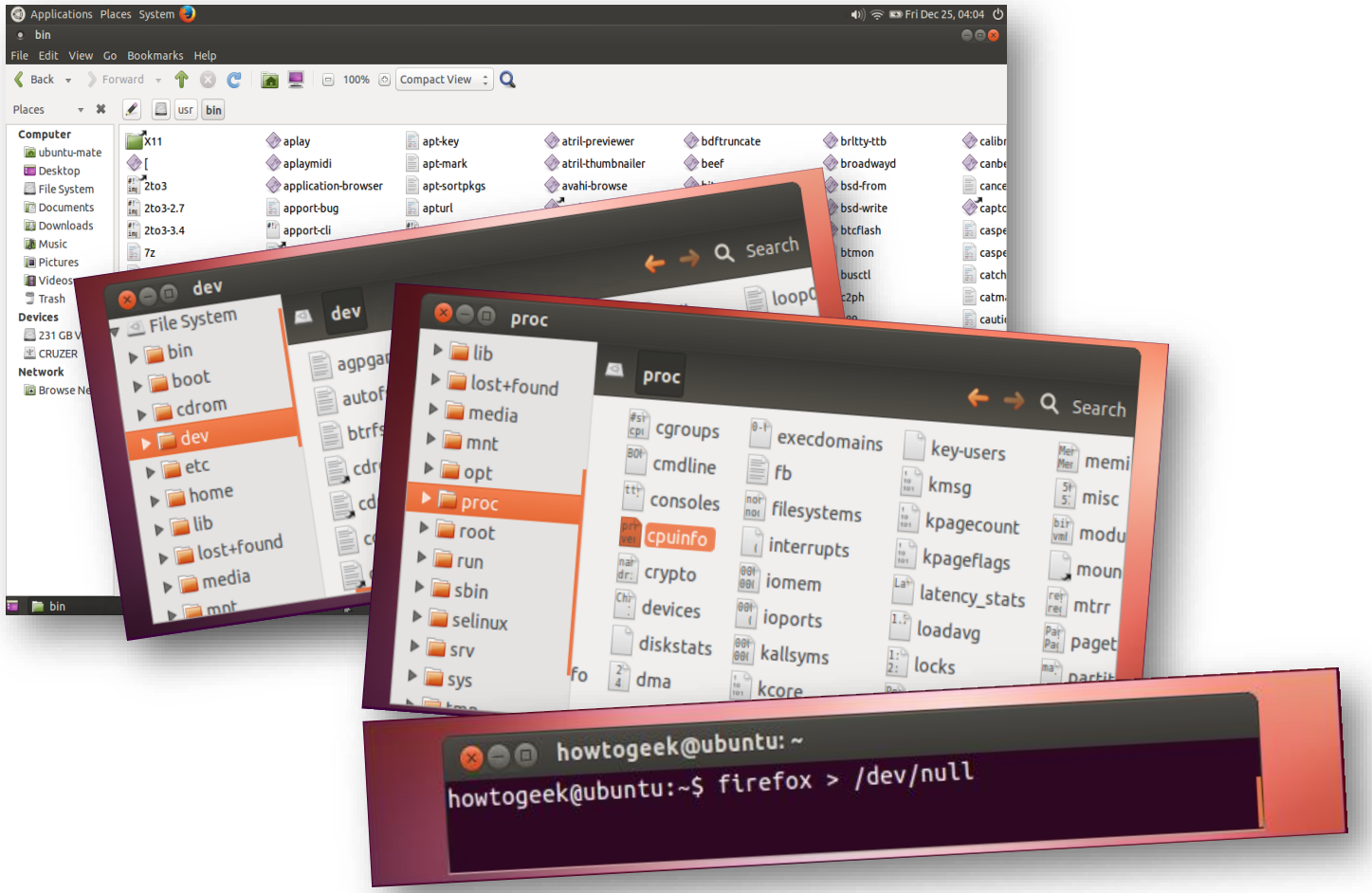
"Everything is a file"

- Um dos princípios chave do Unix
  - Seguido por muitos SO modernos
  - Algumas excepções (até no Unix)

- Objetos que o SO gere são acessíveis aos processos através de *descritores de ficheiro*
  - Ficheiros, directorias, dispositivos lógicos, canais de comunicação, etc.

- Vantagens para os utilizadores/programadores
  - Modelo de programação comum
  - Modelo de segurança comum

"Everything is a file"



- A interface do sistema de ficheiros é a mais utilizada para interagir com o sistema operativo

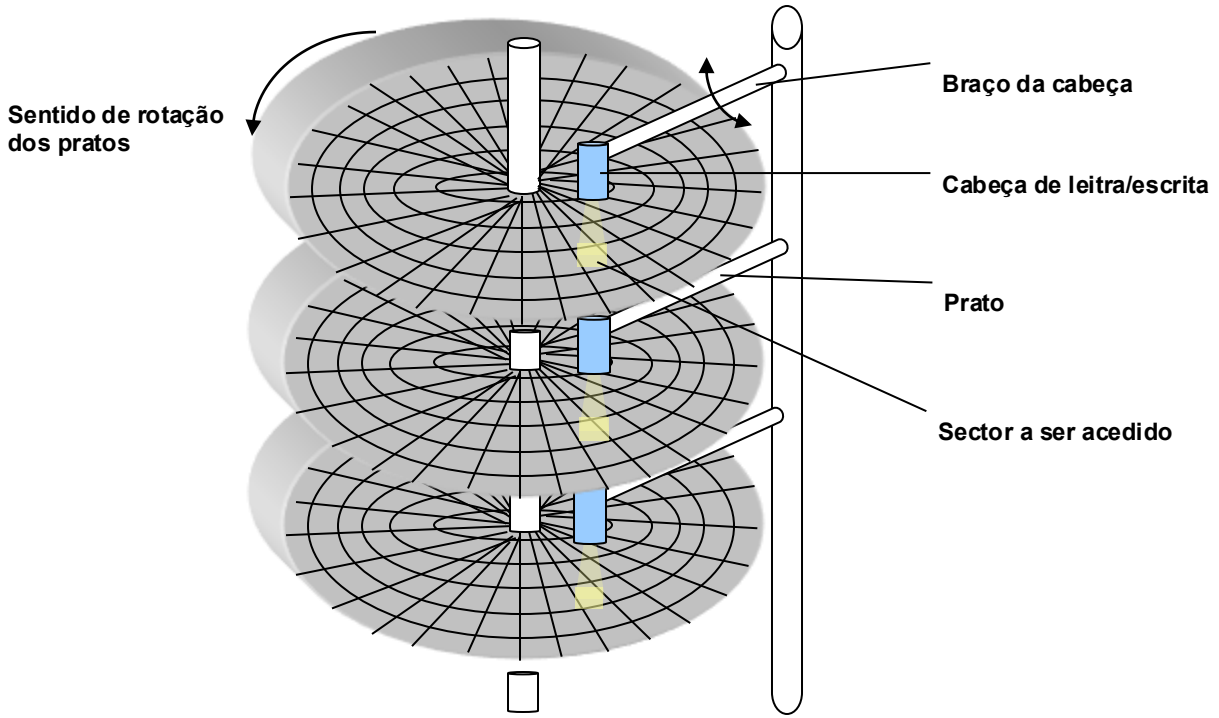
- *Everything is a file* é um dos princípios fundamentais do Unix

Persistência

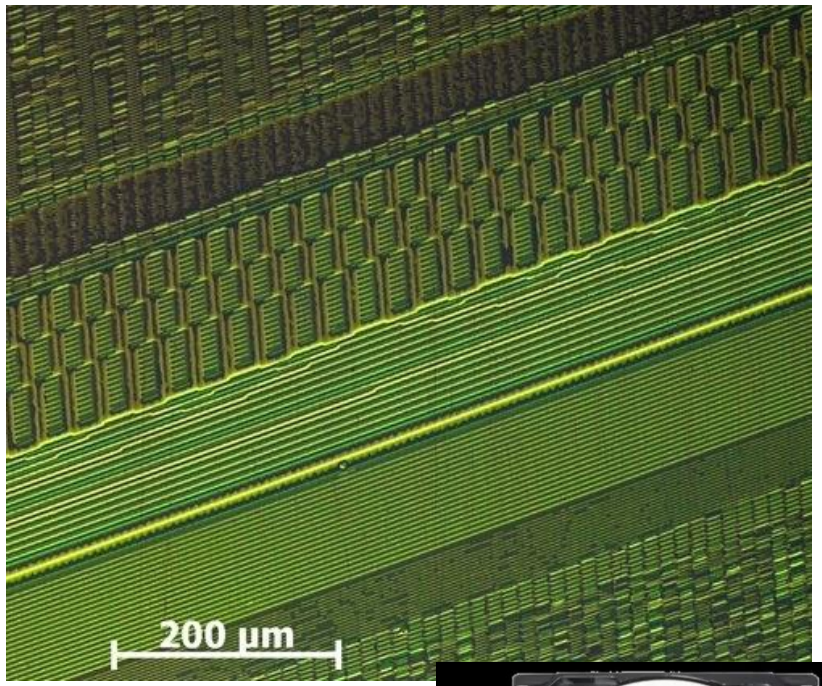
Informação Persistente

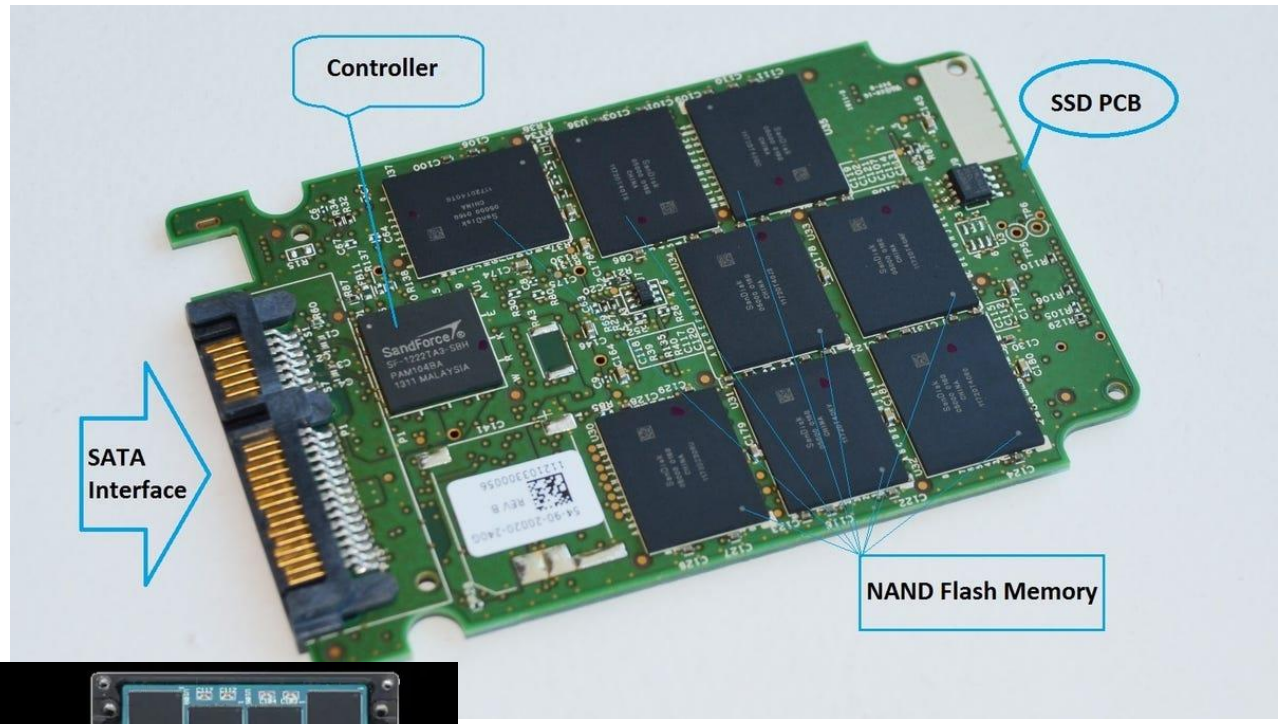
- Uma das características fundamentais e diferente de outras abstrações do Sistema Operativo (processo, espaço de endereçamento,…) é que os ficheiros são **objetos persistentes** ou seja a informação permanece mesmo quando a energia é desligada

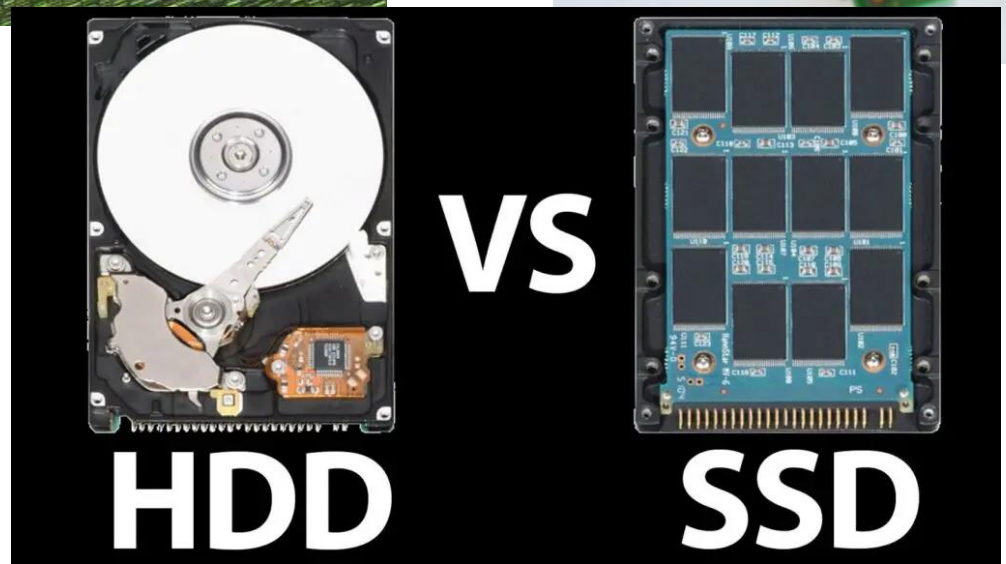
Disco magnético



Diferentes tecnologias em constante evolução

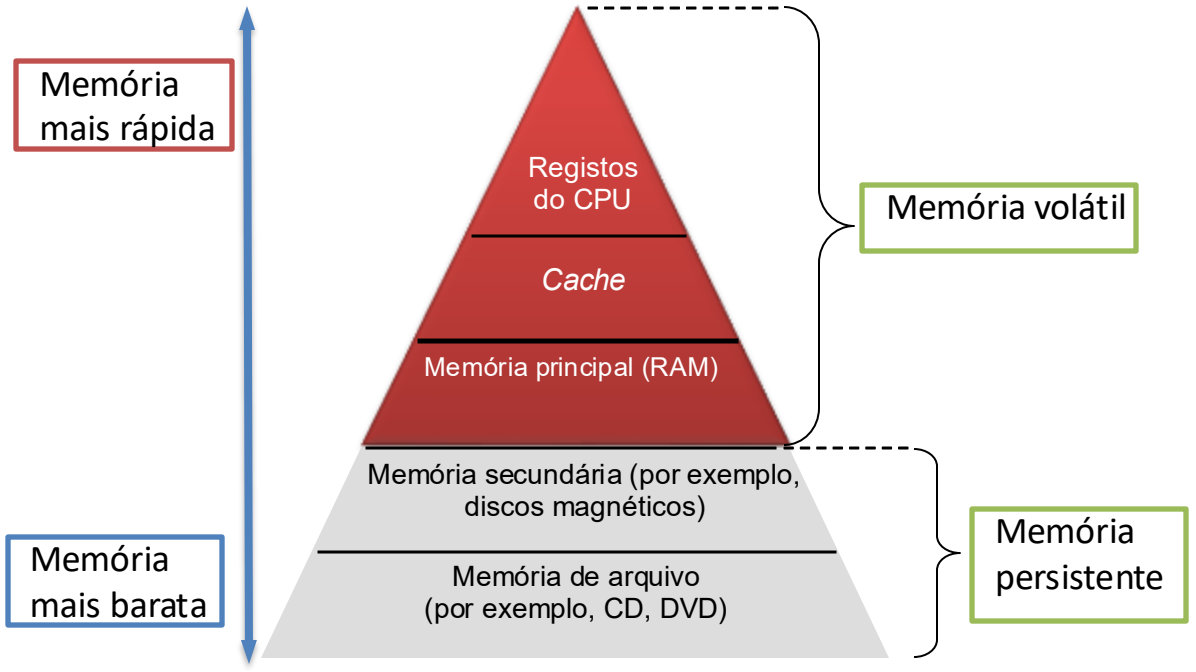






Hierarquia de Memória

- A hierarquia de memória representa normalmente a relação entre memória de acesso rápido, mas de custo elevado versus memórias de custo muito mais reduzido, mas mais lentas
- Outra característica importante é se a memória é volátil ou persistente
- Tirar partido destas diferenças é um exercício de engenharia e constitui um dos problemas que os sistemas operativos tentam resolver



Hierarquia de Memória Simplificada

Consideramos apenas dois níveis

- Memória principal (ou primária):
  - tempo de acesso reduzido
  - custo elevado
  - bom desempenho com acessos aleatórios
  - informação volátil
  - RAM + *caches* [+ registos]

- Memórias secundárias (ou de disco):
  - tempo de acesso elevado
  - custo reduzido
  - pior desempenho com acessos aleatórios (entre blocos diferentes)
  - informação persistente

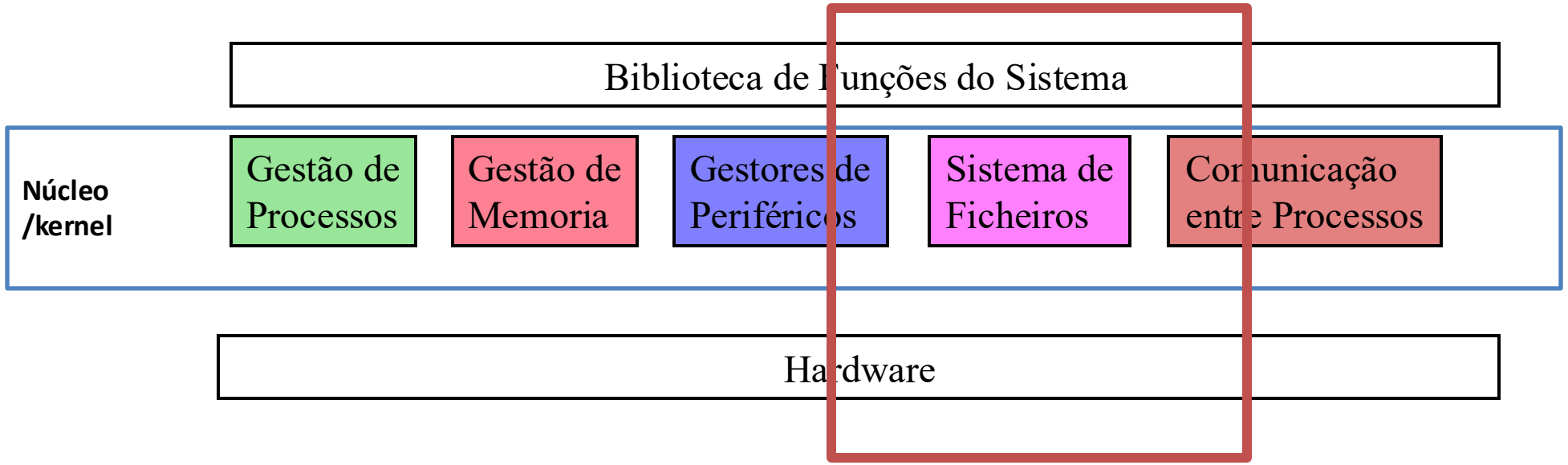
Alguns valores de contextualização

execute typical instruction	1/1,000,000,000 sec = 1 nanosec
fetch from L1 cache memory	0.5 nanosec
branch misprediction	5 nanosec
fetch from L2 cache memory	7 nanosec
Mutex lock/unlock	25 nanosec
fetch from main memory	100 nanosec
send 2K bytes over 1Gbps network	20,000 nanosec
read 1MB sequentially from memory	250,000 nanosec
fetch from new disk location (seek)	8,000,000 nanosec
read 1MB sequentially from disk	20,000,000 nanosec
send packet US to Europe and back	150 milliseconds = 150,000,000 nanosec

Sistemas de Gestão de Ficheiro

Objetivo do sistema de gestão ficheiros:

Virtualizar os dispositivos de armazenamento da informação persistente de forma a que utilizadores e programadores utilizem ficheiros e diretorias



Virtualização da Memória Persistente

- Algumas semelhanças com a virtualização da memória principal (RAM), mas com diferenças fundamentais:
  - Persistência da informação
  - Tempos de leitura e escrita elevados
  - Dimensão da informação guardada
  - Partilha de informação muito mais frequente
  - Segurança associada à partilha

Objeto Ficheiro

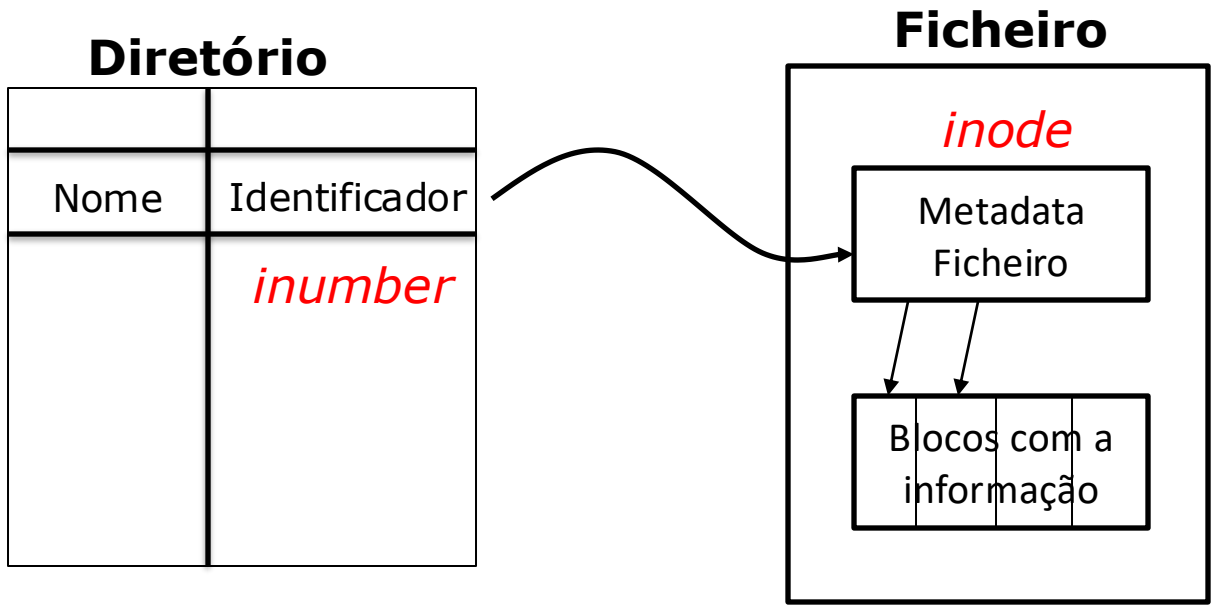
UM FICHEIRO É UMA COLEÇÃO DE DADOS PERSISTENTES, GERALMENTE RELACIONADOS, IDENTIFICADOS POR UM NOME

- A estrutura de dados é normalmente um **vetor de bytes**
  - O SO não conhece a organização interna dos ficheiros que depende das aplicações que os usam
- Tem um **nome**
  - Como é habitual nos objetos do sistema operativo existem **dois** identificadores um para uso dos utilizadores (cadeia de caracteres) e outro interno para o SO (inteiro) que o identifica nas operações internas
- O modo de acesso mais frequente é sequencial pelo que os ficheiros têm associado um índex que indica a posição corrente onde se está a ler ou escrever (normalmente designada ***offset***)
- Tem **informação de gestão** – *metadata* – dono, data de criação, privilégios, etc.

Objeto Diretório

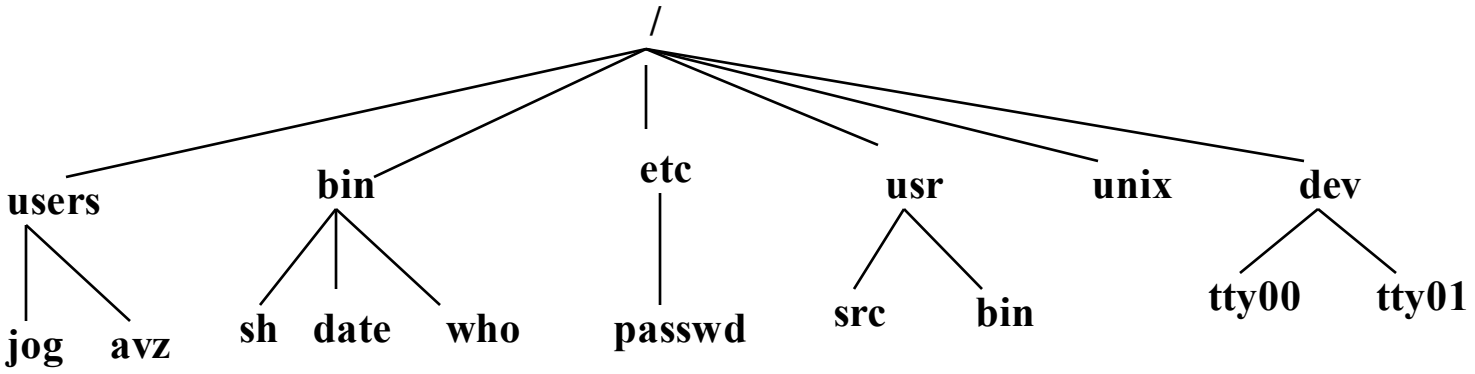
UM DIRETÓRIO É UMA LISTA DE FICHEIROS E DE ATRIBUTOS ASSOCIADOS AOS FICHEIROS

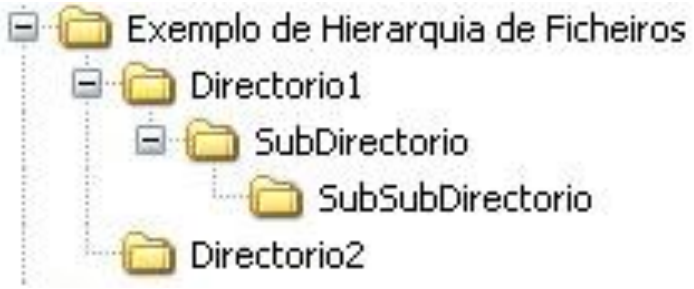
- Lista dos nomes dos ficheiros
- No Unix a lista apenas mantem a relação entre nome (cadeia de carateres) e indentificador interno, noutros sistemas de ficheiros pode ter metadata



Árvore de Diretórios

- Para simplificar a organização da informação, os diretórios estão normalmente relacionados de uma forma hierárquica
- A navegação nas arvores dos diretórios é intuitiva





Caminhos de Acesso – *Pathname*

- A hierarquia de diretórios implica regras para designar os ficheiros

- Nomes absolutos:
  - caminho de acesso desde a raiz
  - Ex.: `/home/joao/SO/project.zip`

Mas ter de fornecer sempre o nome absoluto de um ficheiro é fastidioso e pouco flexível...

- Nomes relativos:
  - caminho de acesso a partir do diretório corrente
  - diretório corrente mantido para cada processo como parte do seu contexto:

`./SO/project.zip`
(supondo que o diretório corrente é **`/home/joao`**)

`../SO/project.zip`
(supondo que o dir. corrente seja **`/home/joao/teo`**)

Nomes e Extensões

- Os nomes de ficheiros tem uma extensão introduzida por um “.”

- Alguns sistemas, como o Unix, não atribuem qualquer significado aos nomes e extensões:
  - As extensões são **apenas convenções mantidas pelos utilizadores e pelas ferramentas** que trabalham sobre os ficheiros.
  - o compilador de C espera que o código fonte esteja num ficheiro com a extensão `.c` e produz um `.o`, uma imagem pode ter extensão `.jpg`, um ficheiro de música `.mp3`

- Em Windows, as extensões podem ou não ser obrigatórias dependendo do sistema de ficheiros utilizado:
  - No FAT a extensão é obrigatória e tem no máximo três caracteres (como no MS/DOS)
  - No NTFS a extensão não é obrigatória e o carácter “.” é interpretado como outro carácter.

Atributos de um Ficheiro

- Para além do tipo, a meta-informação do ficheiro possui usualmente os seguintes atributos:
  - Protecção
    - quem pode aceder ao ficheiro e quais as operações que pode realizar.
  - Identificação do dono do ficheiro
    - geralmente quem o criou.
  - Dimensão do ficheiro
  - Data de criação, última leitura e última escrita

Ficheiros em Unix

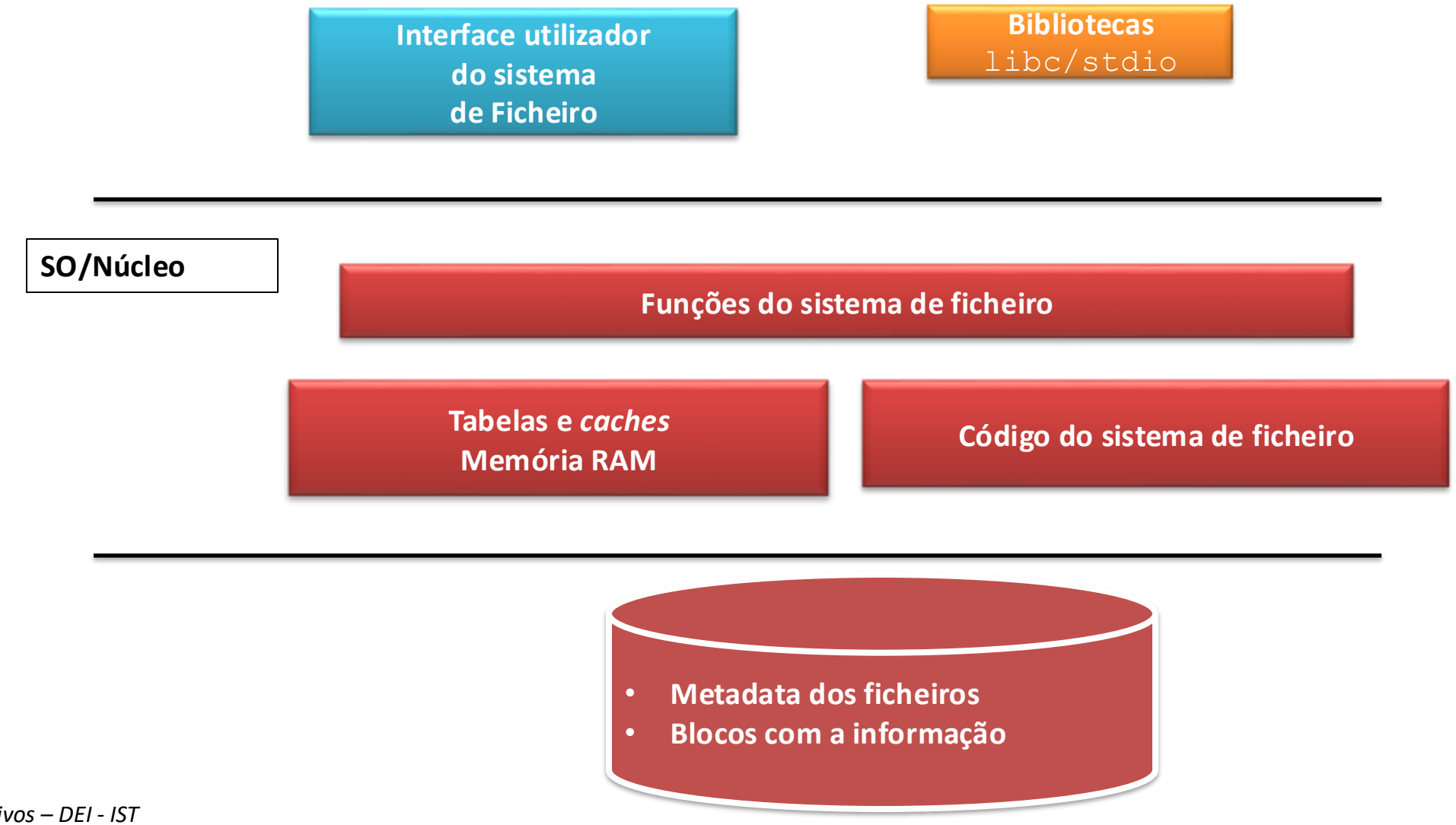
- Tipos de ficheiros:
  - Normais – sequência de octetos (*bytes*) sem uma organização em registos
  - Ficheiros especiais – periféricos de E/S, *pipes*, FIFOS, *sockets*
  - Ficheiros directório

- Quando um processo se começa a executar o sistema abre três ficheiros especiais
  - `stdin` – *input* para o processo (fd – 0)
  - `stdout` – *output* para o processo (fd – 1)
  - `stderr` – saída para assinalar erros (fd – 2)

File Descriptor

- Objetos do sistema de ficheiros são acessíveis aos processos através de **descritores de ficheiro** – *file descriptor*
- Os valores são inteiros que variam de zero até um valor máximo dependente do sistema
- Um conjunto de *system calls,* em grande medida padronizadas, tem como parâmetro um *file descriptor* e permite executar sobre os diversos tipos de ficheiros as operações CRUD e de gestão
- Vantagens para os utilizadores/programadores
  - Modelo de programação comum a muitos dos objetos do SO
  - Modelo de segurança comum
- Um dos princípios chave do Unix, depois seguido pela maioria dos SO

Organização do Sistema de Gestão de Ficheiros



Problemas a Resolver no desenho de um SGF

- Eficiência
  - Fazer com que as funções mais utilizadas (sobretudo a leitura, mas também a escrita) sejam eficientes
  - Interação uniforme: os utilizadores não se devem aperceber dos diversos níveis do sistema, sendo desejável um desempenho uniforme para ficheiros grandes ou pequenos, mais ou menor carga do SGF
- Otimização de recursos,
  - Espaço no disco e na memória primária do sistema operativo
- Segurança da informação persistente
- Fiabilidade da informação que não deve ser perdida ou adulterada

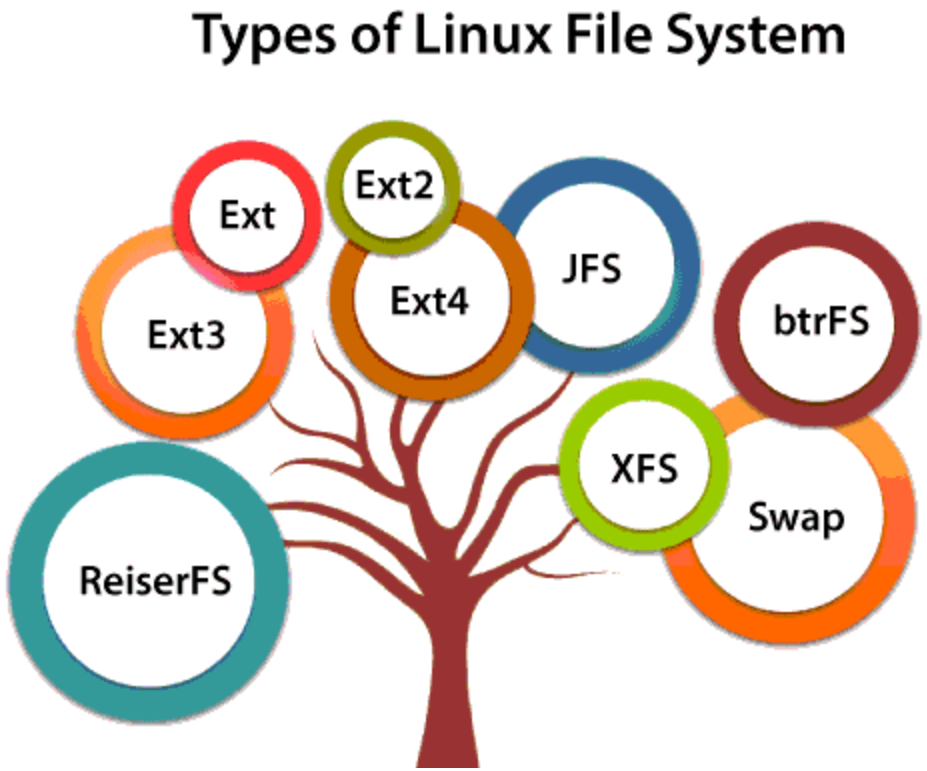
Múltiplos SGF em SO diferentes

- CP/M – Digital Research,
- FAT – Microsoft,
- NFST – Microsoft,
- HTFS – OS/2,
- HFS+ - Apple, etc…

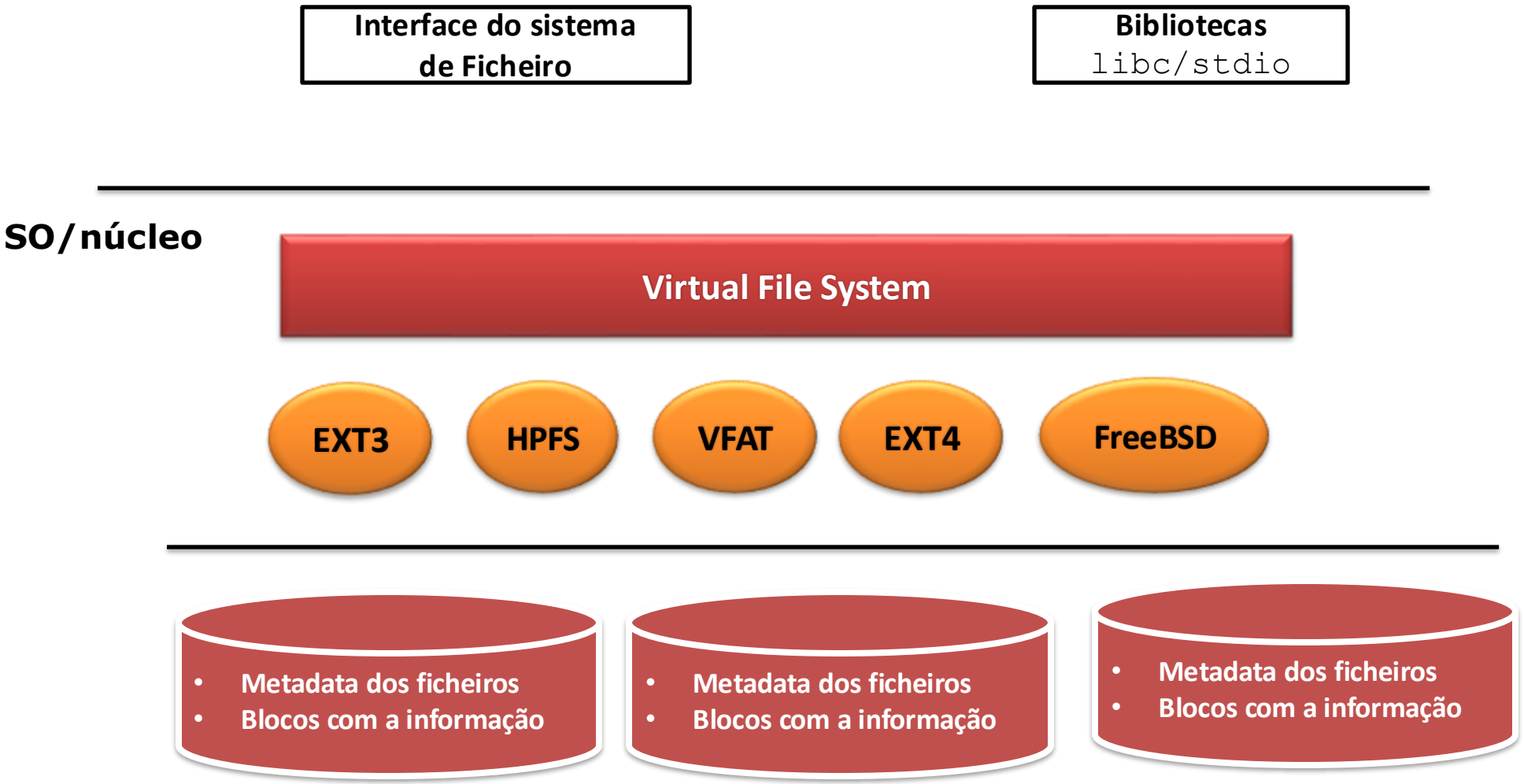
- No Unix/Linux também existem diversas alternativas
- O sistema de ficheiros do Linux teve a sua origem na versão 7 do *Unix File System* (UFS V7) e que continuou inalterada até ao Unix System V.
- O *Berkley Fast File System* (FFS) alterou o formato dos diretórios de modo a permitir ficheiros com nomes até 255 caracteres e propôs uma nova organização da informação em disco que o tornou mais eficiente.

Evolução dos Sistemas de Ficheiros Linux

- A primeira evolução do sistema de ficheiros do Linux deu-se com a introdução do sistema de ficheiros ***extend*** **(Ext)** que incorporava as propostas do *Berkley Fast File System.*
- A segunda evolução, mais significativa, introduziu o sistema de ficheiros ***extend 2*** (**Ext2**) e o ***Virtual File System*** (**VFS**).
- O VFS é um conjunto de estruturas em memória que permite ao sistema Linux suportar em simultâneo várias partições com sistemas de ficheiros diferentes, sendo possível suportar partições com FAT, outras Ext2, etc.
- A evolução seguinte, consistiu no desenvolvimento do Ext3 - o sistema de ficheiros Ext2 com estruturas auxiliares para assegurar a consistência do sistema de ficheiros em caso de faltas - *journaling*.



Organização VFS



Conclusões

- Os objetos **ficheiro** e **diretório** permitem-nos virtualizar o funcionamento de todos os mecanismos de armazenamento persistente da informação
- O **Sistema de Gestão de Ficheiros** implemente esta virtualização. Ao longo dos tempos houve muitas versões de SGF, em particular no Unix/Linux
- No Linux, o *Virtual File System* é uma abstração que permite com a mesma interface utilizar diferentes SGF
- A interface programática baseia-se no conceito de um ficheiro aberto referenciado por um *file descriptor* sobre o qual se executam um conjunto vasto de *system calls*