

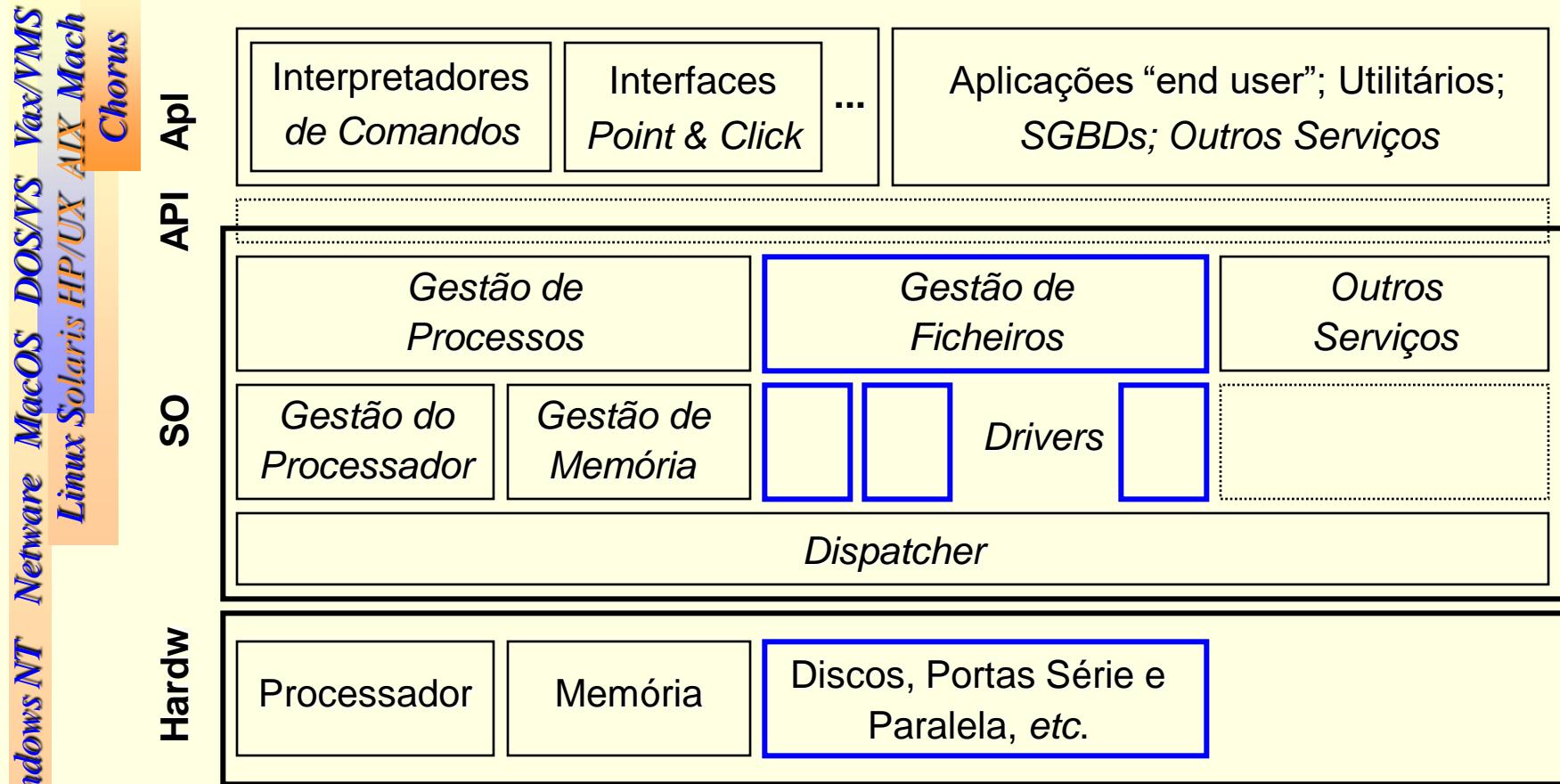
# *Fundamentos de Sistemas de Operação*

Unix Windows NT Netware Mac OS DOS/V/S Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus

## *Gestão de Ficheiros*

### a) Conceitos [Revisão]

# O Sistema de Ficheiros



# *Ficheiro: o Conceito*

## □ O conceito de Ficheiro

- O conceito pode ser simples, mas a sua implementação pelos diferentes SO's é muito diversa
- Uma definição sucinta:
  - Entidade, referenciável por um <nome>, usada para armazenar dados, de uma forma persistente
  - Estrutura de dados, armazena 0 ... n ocorrências de um dado tipo de Registo
- Operações elementares:
  - Criar ou Apagar um ficheiro; Abrir ou Fechar um ficheiro
  - Ler ou Escrever um “registo” num ficheiro

# *Ficheiro: o Conceito*

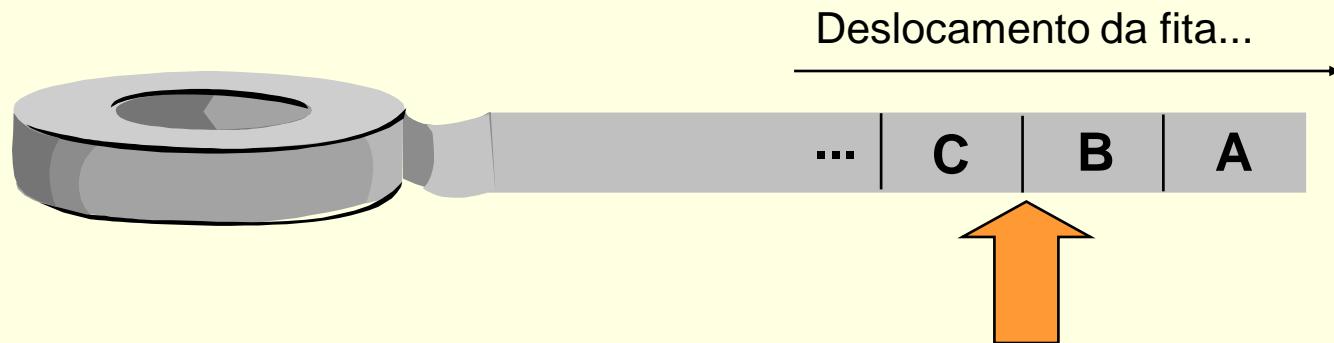
## □ O conceito de Ficheiro (continuação)

- Foi fortemente influenciado pelas “necessidades” das Linguagens de Programação (e influenciou-as):
  - Ficheiros sequenciais, directos e indexados
  - Ficheiros “de texto”
  - Registos de comprimento fixo e variável (um “registo” pode ser uma struct - sempre da mesma dimensão - uma union (struct variável) - etc.)
- O “conceito minimalista” de ficheiro:
  - *Byte Stream*: um ficheiro é uma sequência (eventualmente não-estruturada) de *bytes* - ou “quem o escreveu que o leia”

# Ficheiro: o Conceito

## □ O conceito de Ficheiro (continuação)

- O *file pointer* (emulação da noção de cabeça de leitura)

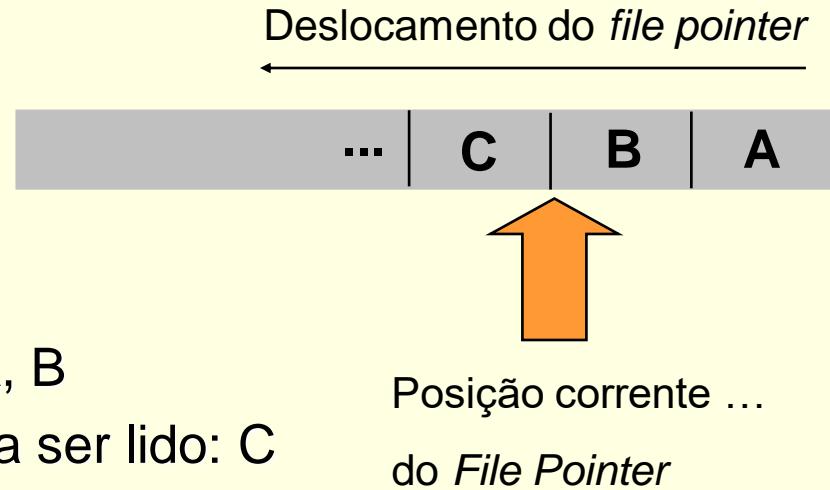


- “Registros” lidos: A, B
- Próximo “registro” a ser lido: C

# *File Pointer: o Conceito*

## □ O conceito de *File Pointer*

- O ficheiro “está imóvel” e o *file pointer* desloca-se

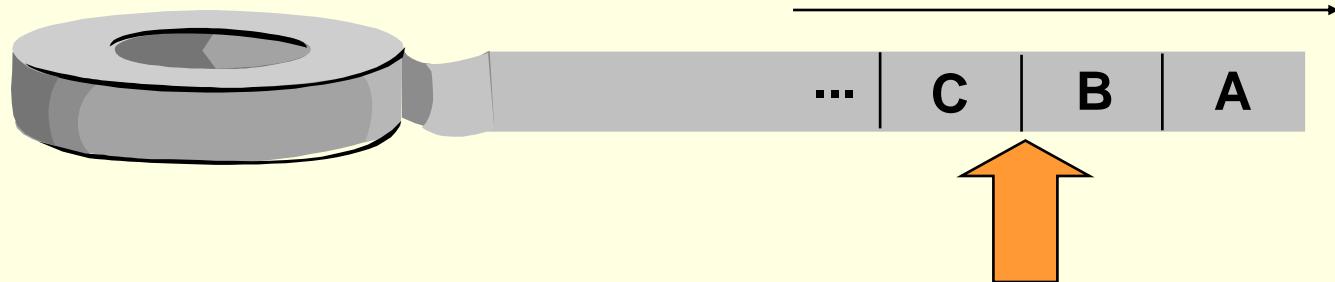


- “Registros” lidos: A, B
- Próximo “registro” a ser lido: C

# *Ficheiro: Organização e Acesso*

## □ Organização Sequencial

- Os dados estão registados em posições fisicamente contíguas, e não é possível aceder à posição  $n$  sem ter “passado por cima” das  $n-1$  anteriores...

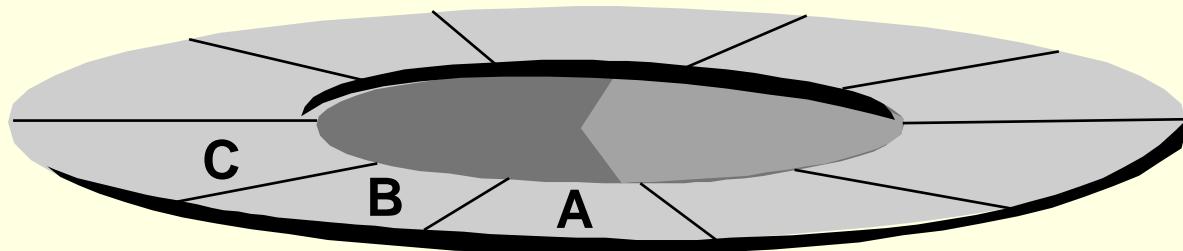


Numa “fita magnética” não é possível organizar os dados de outra forma...  
o **acesso** tem de ser sequencial

# *Ficheiro: Organização e Acesso*

## □ Organização Directa

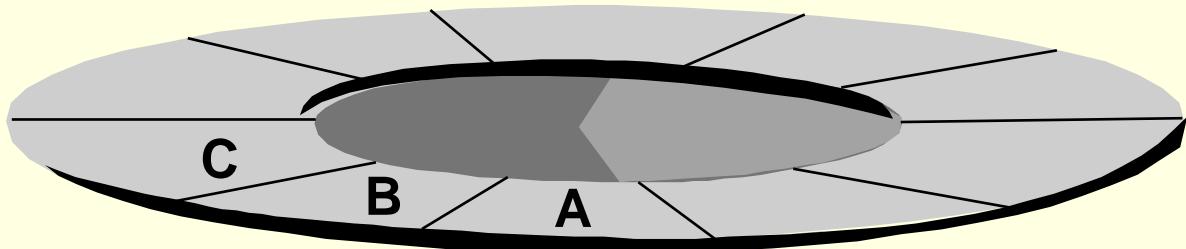
- O dispositivo permite o acesso directo aos dados - i.e., é possível aceder-se à posição  $n$  sem ter “passado por cima” das  $n-1$  anteriores...



# *Ficheiro: Organização e Acesso*

## □ Organização Directa: Acesso Directo vs. Sequencial

- Se dispositivo permite o acesso directo aos dados, então é possível aceder-se directamente a um “registo” - desde que o Sistema de Ficheiros saiba como “calcular” a posição física correspondente ao item desejado...

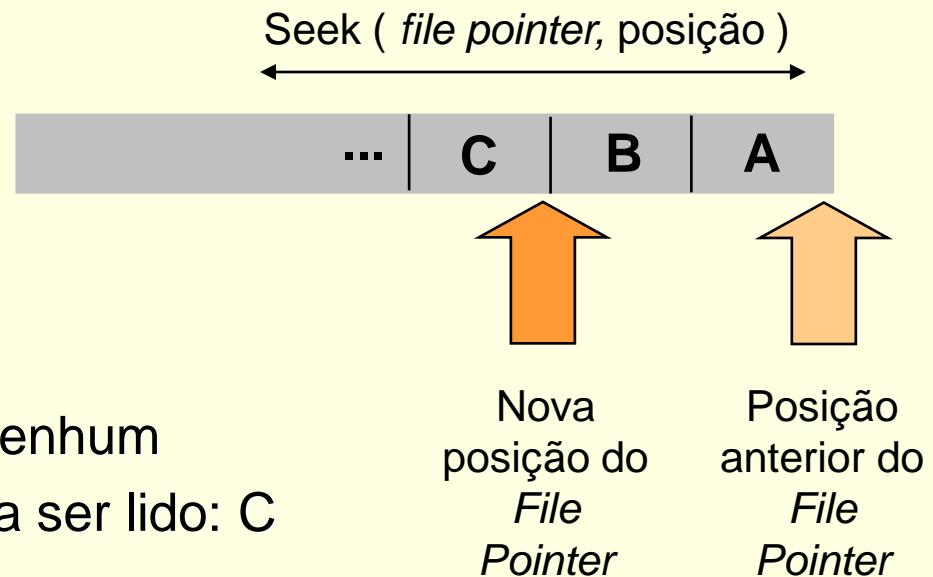


- Claro que continua a ser possível o acesso sequencial...

# O File Pointer e os Métodos de Acesso

## □ O File Pointer e o Acesso Directo

- Uma operação para deslocar o *file pointer* sem contudo transferir informação:



- “Registros” lidos: nenhum
- Próximo “registro” a ser lido: C

# Acesso a Ficheiros: Primitivas

## □ Open

- Associa um ficheiro a um identificador que será utilizado em todas as outras primitivas;
- Posiciona o *File Pointer* no início ou, se este for aberto para acrescentar (*append*), no fim do ficheiro;

## □ Close

- Dissocia o ficheiro do identificador; este não pode mais ser usado;
- Garante que, se o ficheiro não está aberto por outros processos, toda a informação que possa existir em *buffers* é despejada (*flushed*) no suporte físico.

# Acesso a Ficheiros: Primitivas

## □ Read/Write

- Lê/Escrive um objecto (um registo, um certo número de *bytes*) a partir da posição corrente do *file pointer*, actualizando-o depois

## □ Seek

- Re-posiciona o file pointer no local indicado:
  - Princípio ou fim do ficheiro, ou
  - Deslocamento (*offset*) a partir de um ponto de referência

# *Sistema de Ficheiros: a Implementação*

## □ A implementação...

- O ficheiro nos Sistemas de Ficheiros:
  - Regras para o nome (quantos e que caracteres, etc.)
  - Tipos de ficheiros (Existem? Quantos? São usados pelo SO ou apenas servem para os utilizadores “verem”?)
  - Extensões (sim ou não? se sim, obrigatórias ou opcionais? São usadas pelo SO ou apenas servem para os utilizadores “verem”?)
- O “ficheiro minimalista” vs. modelos disponibilizados pelas Linguagens de Programação:
  - Se o Sistema de Ficheiros não oferece os “tipos de ficheiros” necessários para uma dada Linguagem, então estes serão implementados numa Biblioteca de Suporte à Execução (a partir dos “tipos básicos” e primitivas disponibilizadas pelo SO) a ligar (*link*) aos programas que são escritos nessa linguagem.

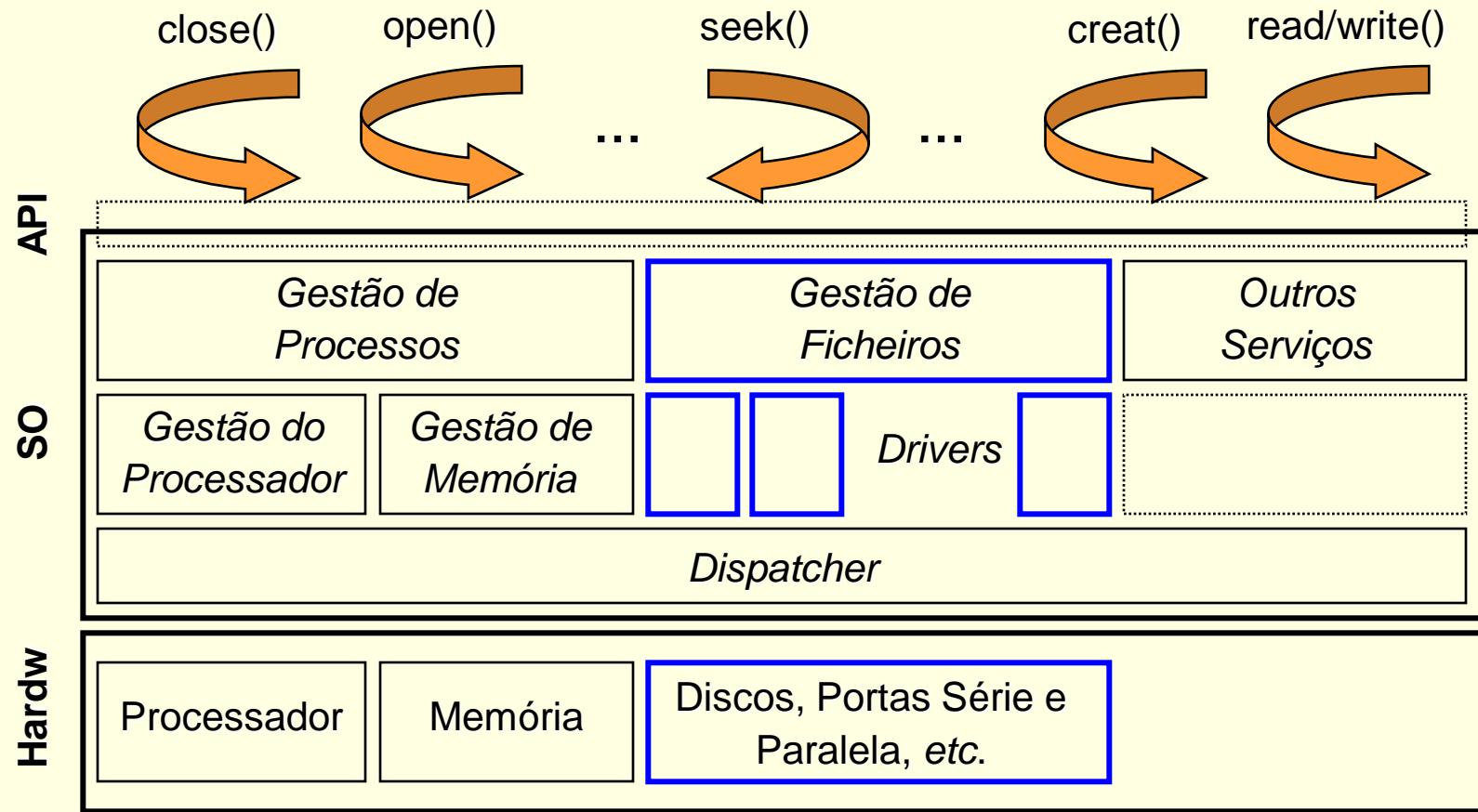
# *Fundamentos de Sistemas de Operação*

Unix Windows NT Netware Mac OS DOS/V/VS Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus

## *Gestão de Ficheiros*

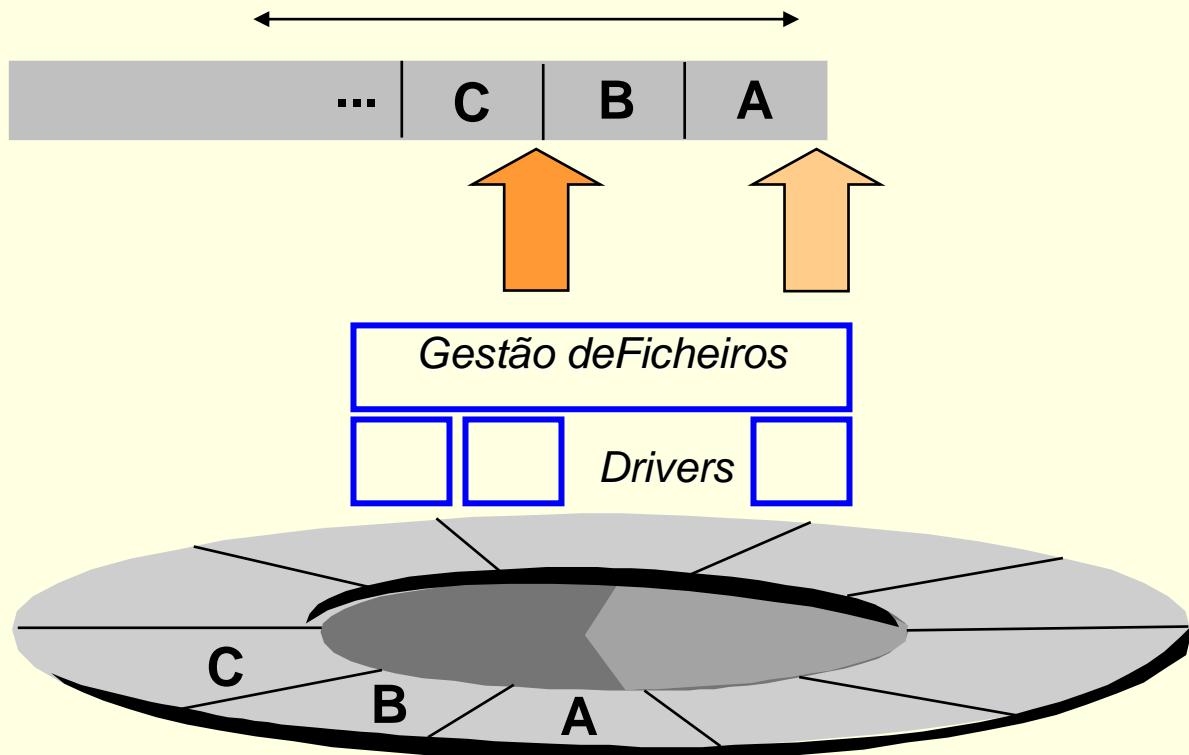
b) Sistema de Ficheiros em Disco –  
Algumas questões...

# O Sistema de Ficheiros



# *Sistemas de Ficheiros: Questões I*

- Como implementar sobre o disco o conceito de ficheiro?



# *Sistemas de Ficheiros: Questões II*

- Como identificar os ficheiros? Que regra para os nomes?
- Como saber onde começa e acaba um ficheiro?
- Como saber que blocos pertencem a um dado ficheiro?
- Dois elementos de dados sequencialmente colocados no ficheiro (lógico) têm de estar em posições contíguas do disco (físico)?

# *Fundamentos de Sistemas de Operação*

Unix Windows NT Netware Mac OS DOS/V/VS Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus

## *Gestão de Ficheiros*

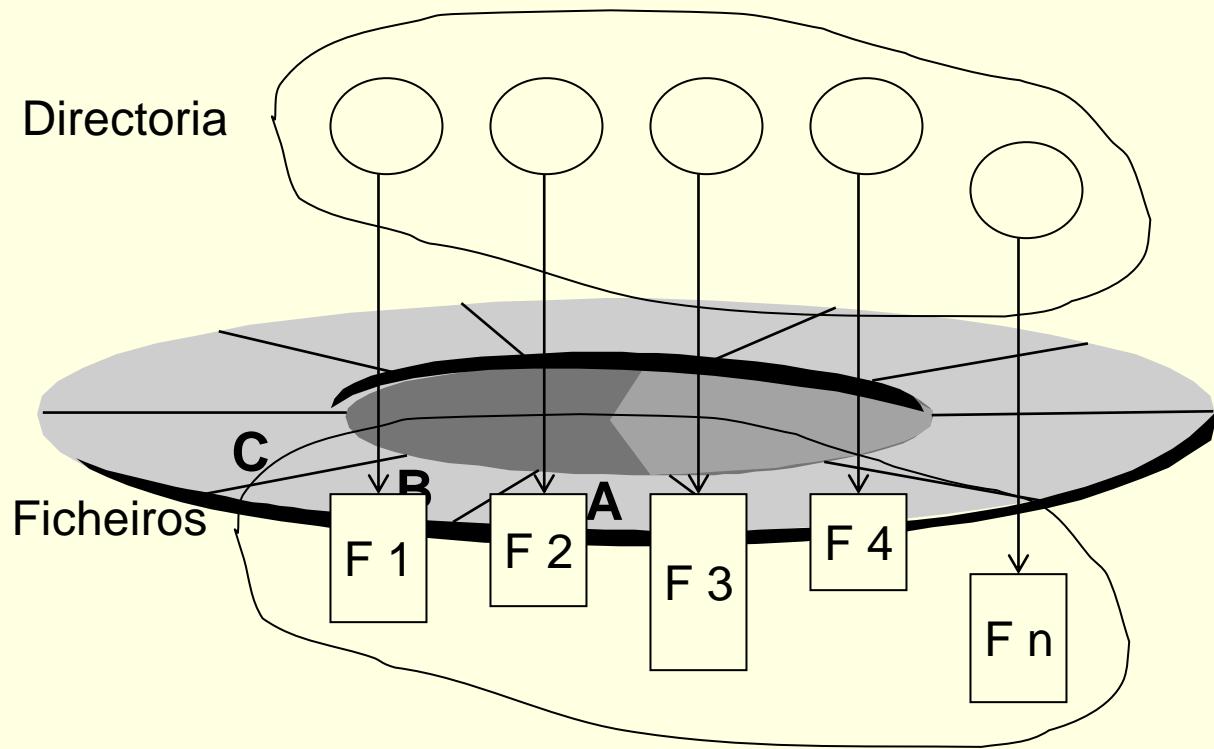
c) Sistema de Ficheiros em Disco –  
O “espaço” de nomes...

# *Sistemas de Ficheiros: Questões*

- Como identificar os ficheiros? Que regra para os nomes?
- Como estruturar o “espaço de nomes”?
  - Um único espaço global “plano” para os nomes?
  - Um espaço independente por utilizador? (e como se partilham ficheiros?)
  - Um espaço hierárquico? (árvore)
  - Um grafo?

# *Estrutura de directorias*

- Colecção de entidades onde é mantida informação sobre todos os ficheiros...



# Estrutura de directorias: Objectivos

- Eficiência – localizar rapidamente um ficheiro.
- Designação (*naming*) – adequada às necessidades dos utilizadores:
  - Dois utilizadores podem dar o mesmo nome a ficheiros diferentes.
  - O mesmo ficheiro pode ter vários nomes diferentes.
- Agrupamento – agrupamento lógico de ficheiros de acordo com as suas propriedades: (por ex., programas em Java, jogos, ...)

# *Directorias: Conteúdo*

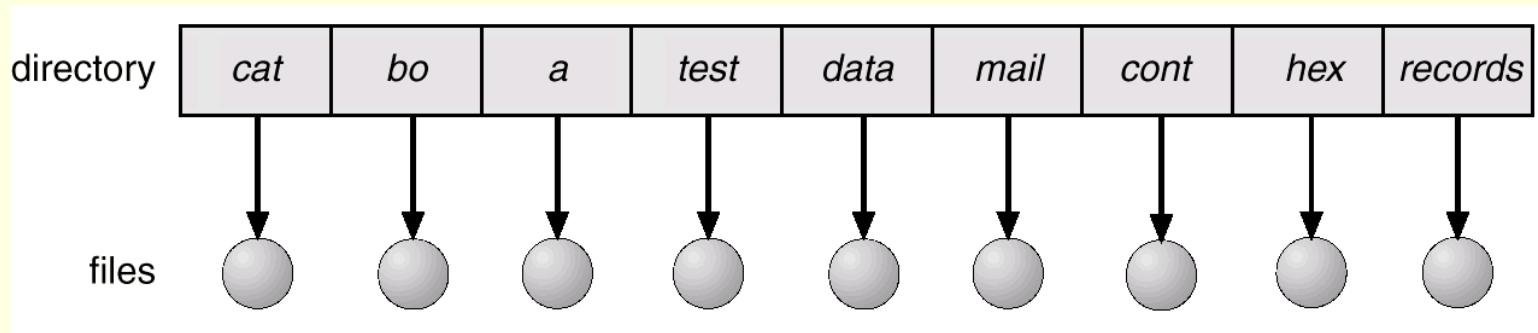
- Para cada “ficheiro”:
  - Nome
  - Tipo
  - Localização
  - Comprimento corrente
  - Data do último acesso (para gestão)
  - Data da última actualização (para backup)
  - ID do dono
  - Informação de protecção (confidencialidade)

# *Directorias: Operações*

- Primitivas sobre o “tipo” Directoria:
  - “Localizar” um ficheiro
  - Criar um ficheiro
  - Apagar um ficheiro
  - Listar o conteúdo (obter lista de ficheiros)
  - Mudar o nome de um ficheiro
  - Navegar na estrutura de directorias

# *Directoria de um nível (única)*

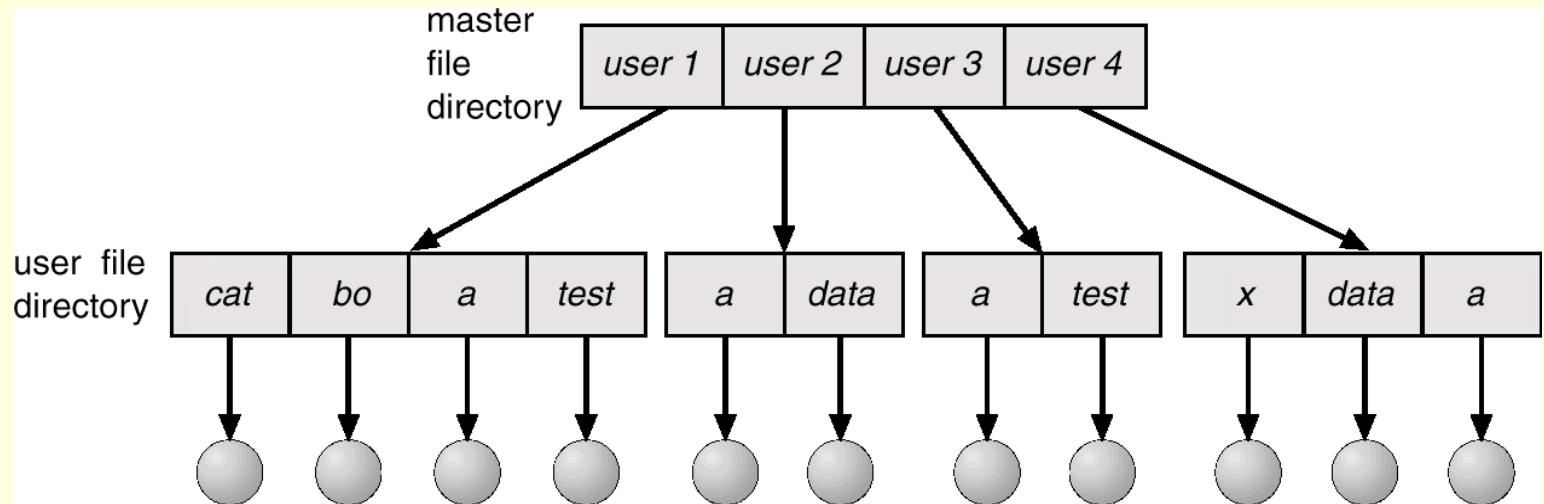
- Uma única Directoria para todos os utilizadores



- Problemas de:
  - Designação: dificuldade de atribuir o nome (colisões)
  - Agrupamento: não é possível

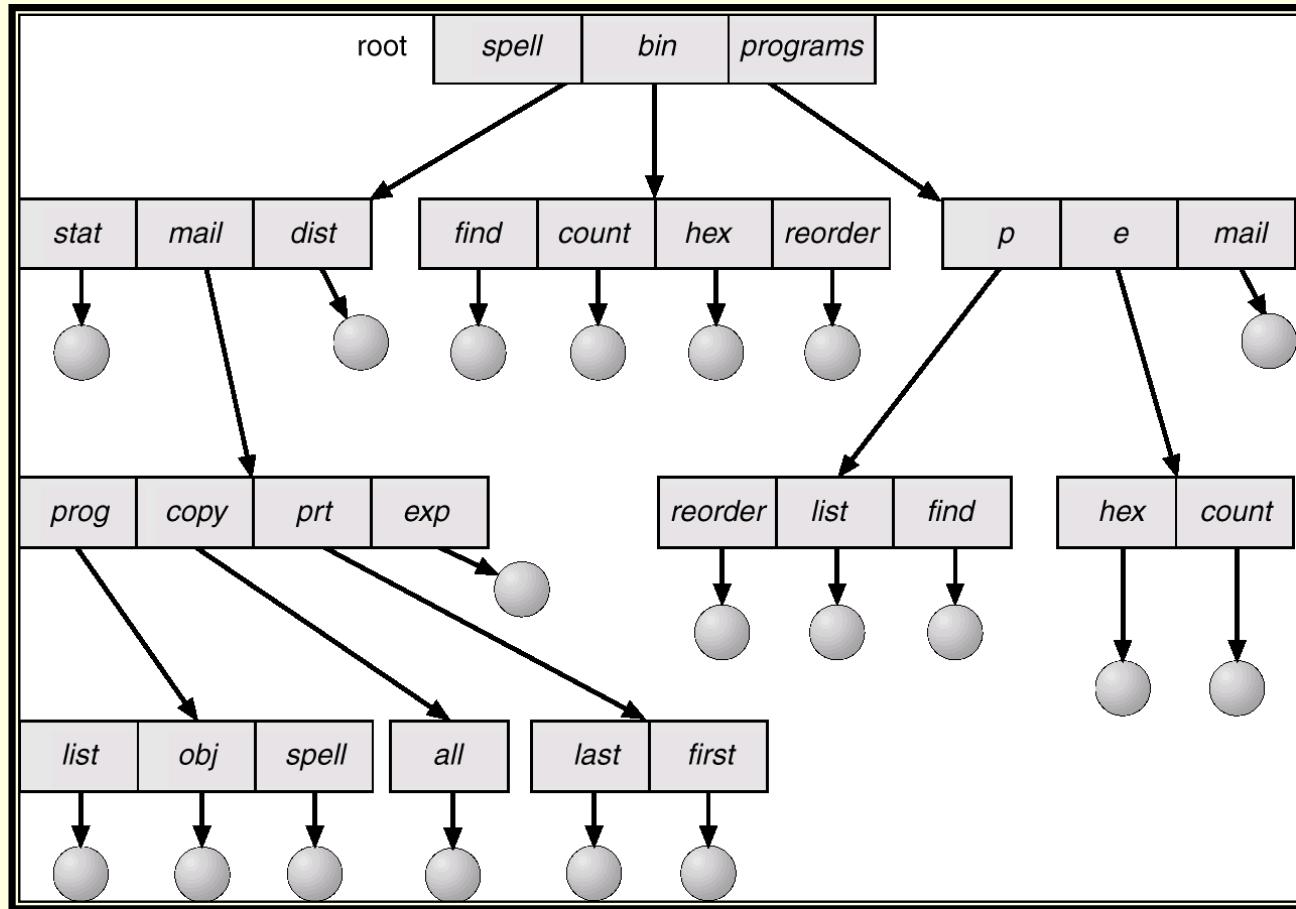
# *Directoria de dois níveis*

- Uma Directoria “privada” para cada utilizador



- Problemas de:
  - Agrupamento (realizado pelo utilizador): não é possível
  - Partilha?

# Directorias em árvore: I



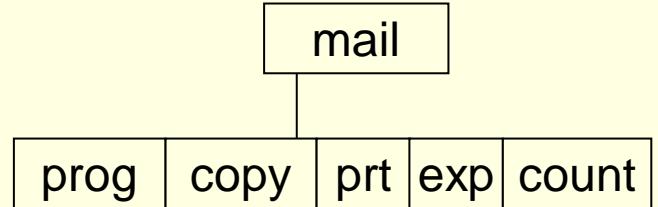
# *Directorias em árvore: II*

- Pesquisa eficiente
- Facilidade de agrupamento
- Directoria corrente (working directory)
  - cd /spell/mail/prog
  - . , ..
- Nomes absolutos (full pathname) e relativos
- Criação e destruição de ficheiros e directorias usando nomes relativos ou absolutos
  - rm <file-name>
  - mkdir <dir-name>

# *Directorias em árvore: III*

- Criação e destruição de ficheiros e directorias: algumas questões, e “cuidados”:

- Se a directoria corrente é /mail, e se cria uma nova directoria
    - mkdir count



- ? Que acontece/deve acontecer se se fizer
    - rm /mail