Laboratório de Pesquisa em Redes e Multimídia

Sistemas de Arquivos

(Aula 23)

Universidade Federal do Espírito Santo
Departamento de Informática

Funções de um SO

- Gerência de processos
- Gerência de memória
- **Gerência de Arquivos**
- Gerência de I/O
- Sistema de Proteção

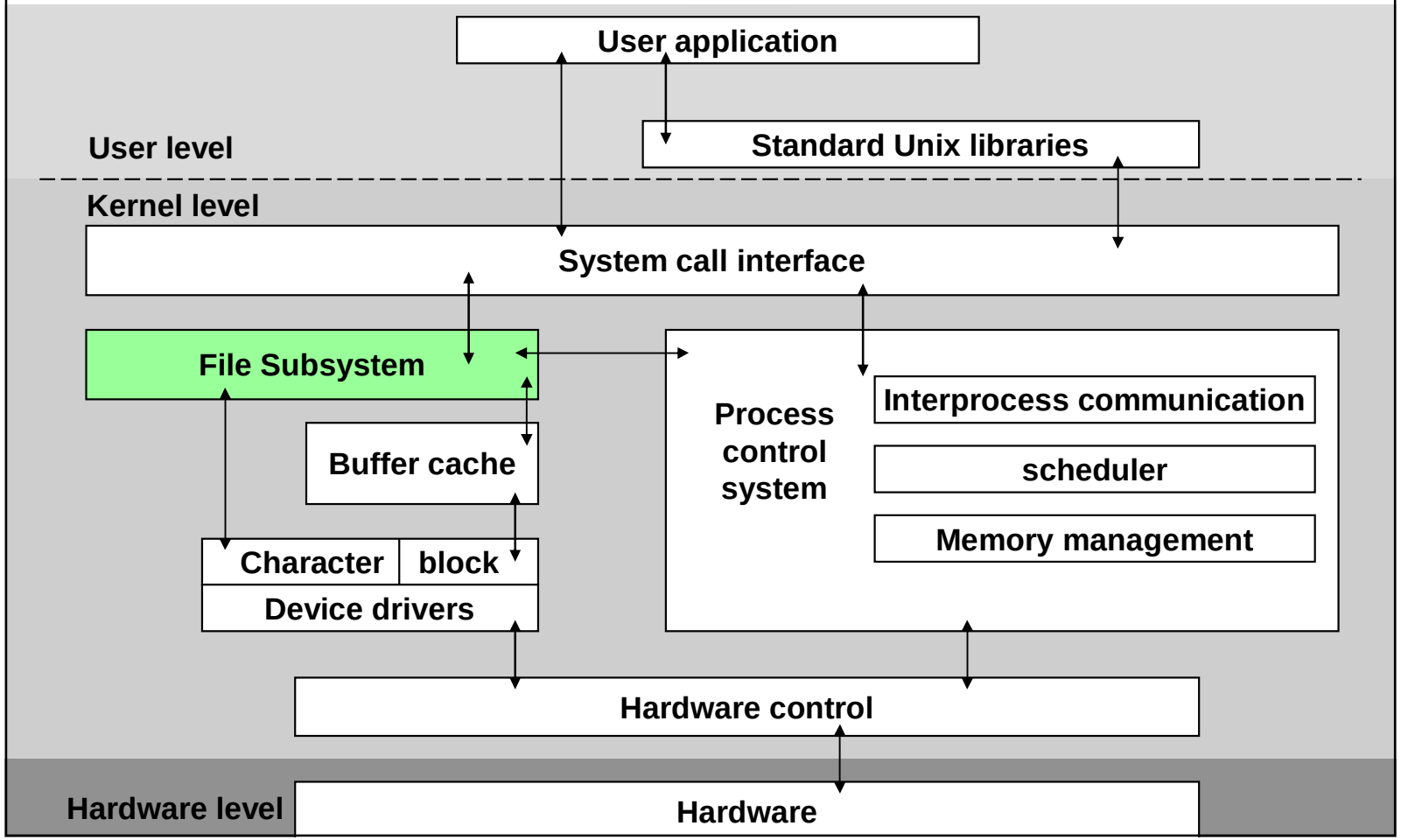
Necessidade de Armazenamento

- Grandes quantidades de informação têm de ser armazenadas
- Informação armazenada tem de sobreviver ao fim do processo que a utiliza
- Múltiplos processos devem poder acessar a informação de um modo concorrente
- ARQUIVO
  - Abstração criada pelo S.O. para gerenciar e representar os dados

Gerência de Arquivos

- Oferece a abstração de arquivos (e diretórios)
- Atividades suportadas
  - Primitivas para manipulação (chamadas de sistema para manipulação de arquivos)
    - criar, deletar
    - abrir, fechar
    - ler, escrever
    - posicionar
  - Mapeamento para memória secundária

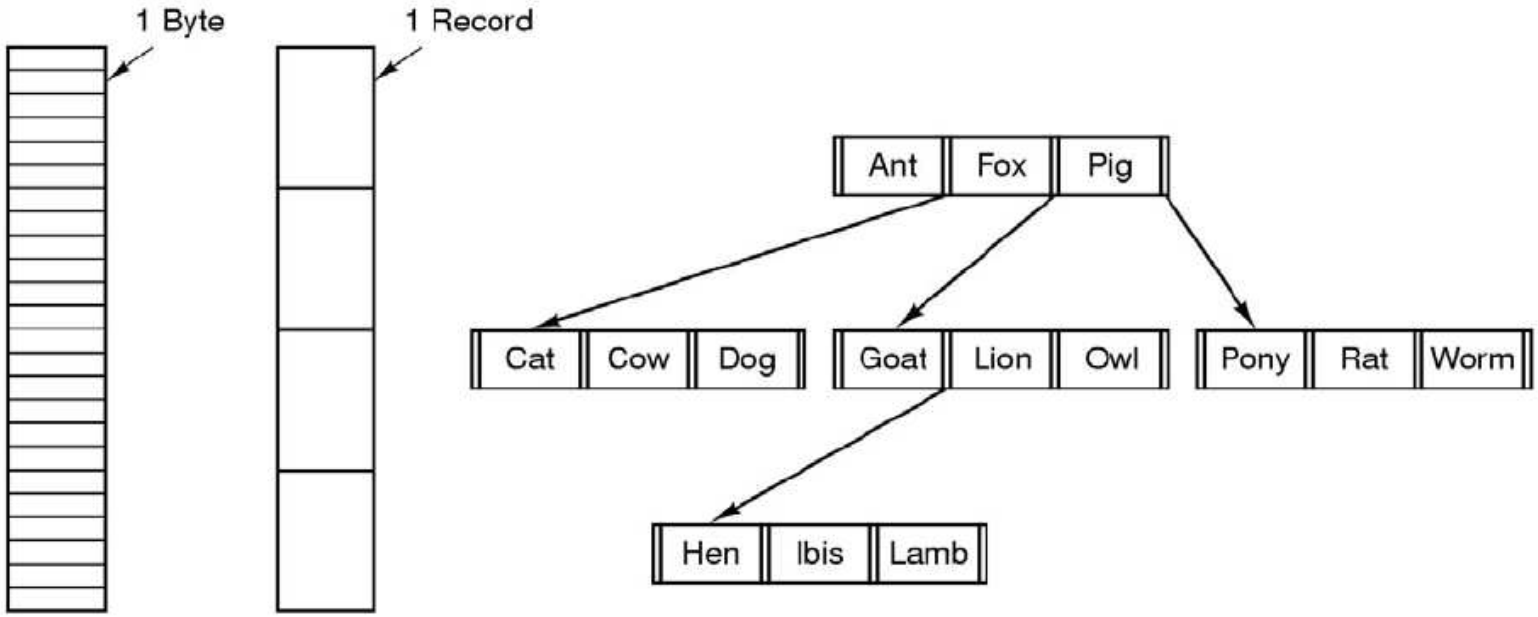
Estrutura Interna do Kernel UNIX



Sistema de Arquivos

- O que é?
  - Um conjunto de arquivos, diretórios, descritores e estruturas de dados auxiliares gerenciados pelo sub sistema de gerência de arquivos
  - Permitem estruturar o armazenamento e a recuperação de dados persistentes em um ou mais dispositivos de memória secundária (discos ou bandas magnéticas)
- Arquivo
  - Um conjunto de dados persistentes, geralmente relacionados, identificado por um nome
  - É composto por:
    - Nome: identifica o ficheiro perante o utilizador
    - Descritor de arquivo: 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

Estrutura Interna de Arquivos (1)



Seqüência não-estruturada de bytes

Seqüência de Registros

Árvore de Registros

Estrutura Interna de Arquivos (2)

- Seqüência não-estruturada de bytes
  - Forma mais simples de organização de arquivos
  - Sistema de arquivos não impõe nenhuma estrutura lógica para os dados, a aplicação deve definir toda a organização
  - Vantagem: flexibilidade para criar estruturas de dados, porém todo o controle de dados é de responsabilidade da aplicação
  - Estratégia adotada tanto pelo UNIX quanto pelo Windows

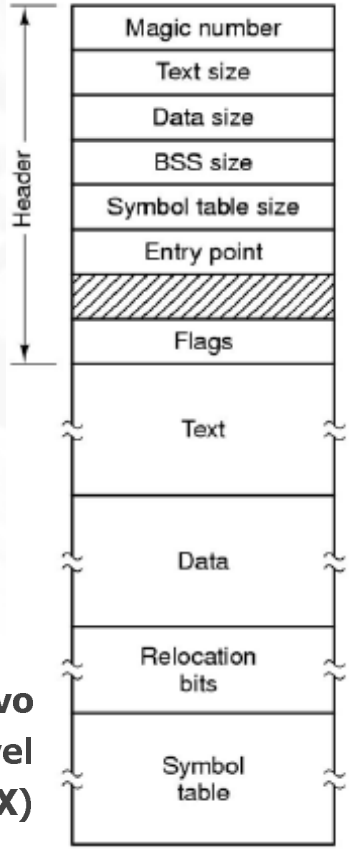
Estrutura Interna de Arquivos (3)

- Seqüência de Registros
  - Em geral, registros de tamanho fixo
  - Operação de leitura retorna um registro
  - Operação de escrita sobrepõe/anexa um registro
- Árvore de Registros
  - Cada registro é associado a uma chave
  - Árvore ordenada pela chave
  - Computadores de grande porte / aplicações que fazem muita leitura aleatória

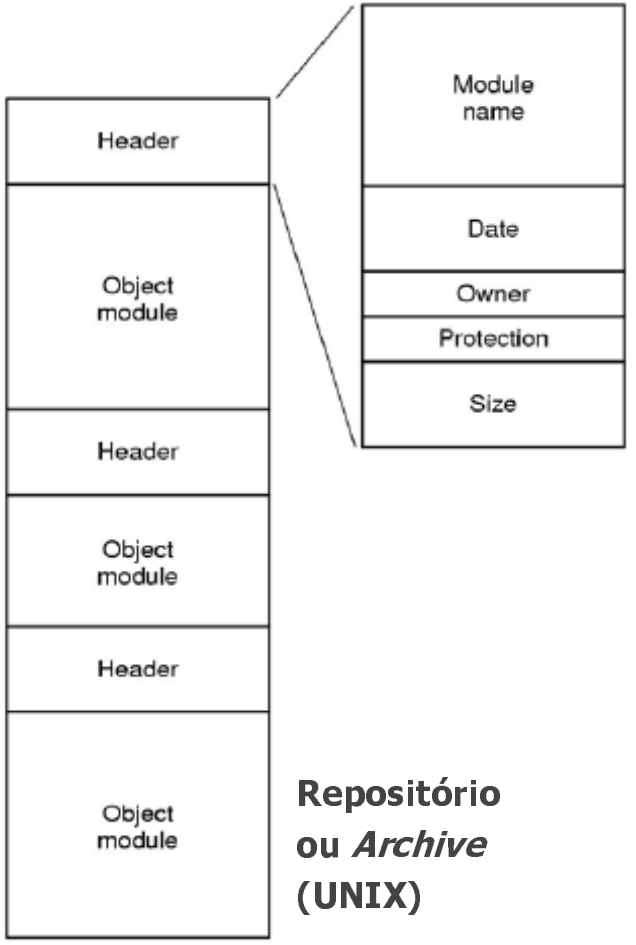
Tipos de Arquivos (1)

- Arquivos Regulares
  - Arquivos ASCII
  - Binários
    - Apresentam uma estrutura interna conhecida pelo S.O.
- Diretórios
  - Arquivos do sistema
  - Mantêm a estrutura do Sistemas de Arquivos

Tipos de Arquivos (2)



Arquivo Executável (UNIX)

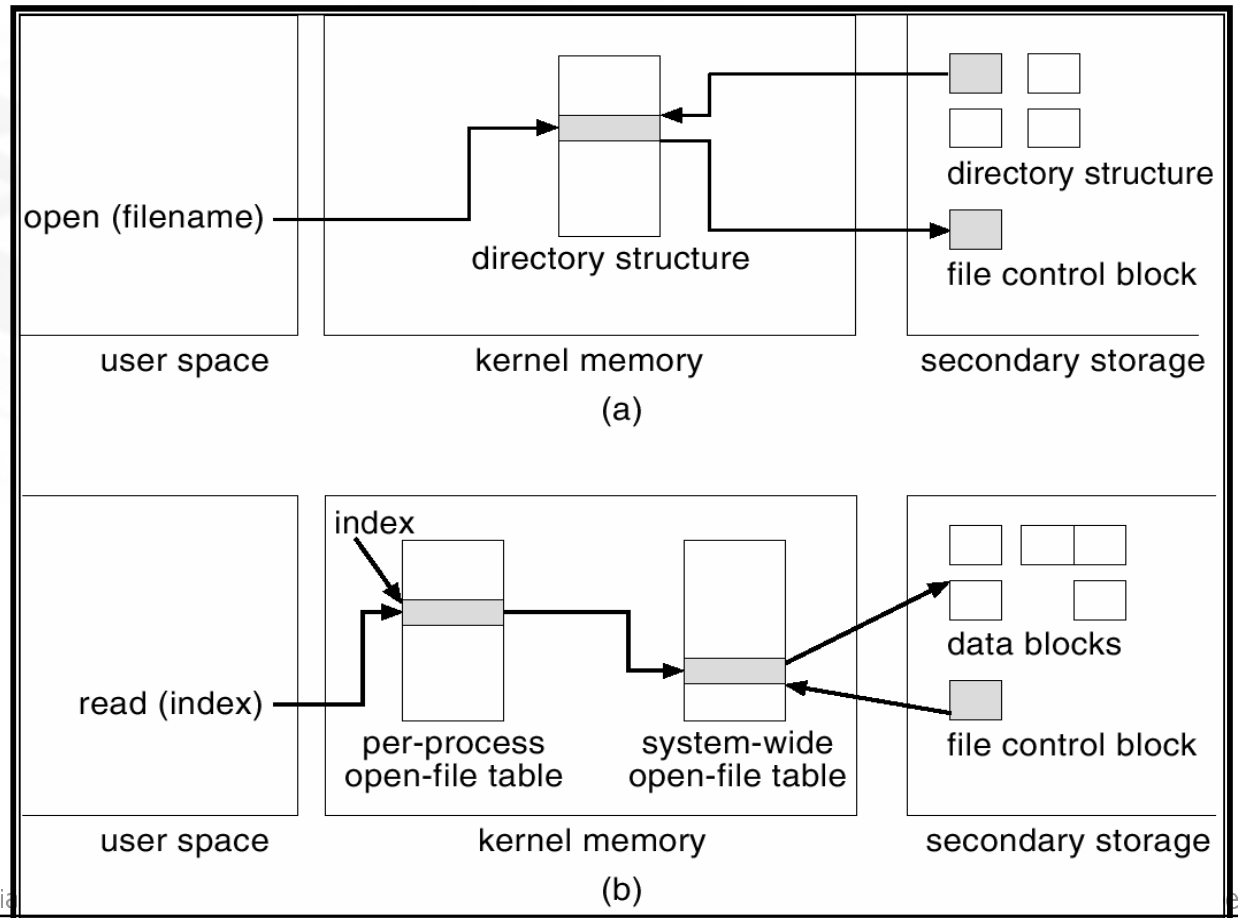


Repositório ou *Archive* (UNIX)

Operações sobre Arquivos

- Dependem do tipo
  - create
  - delete
  - open
  - close
  - read
  - write
  - append
  - seek
  - get attributes
  - set attributes
  - rename

Operações sobre Arquivos



Diretórios (1)

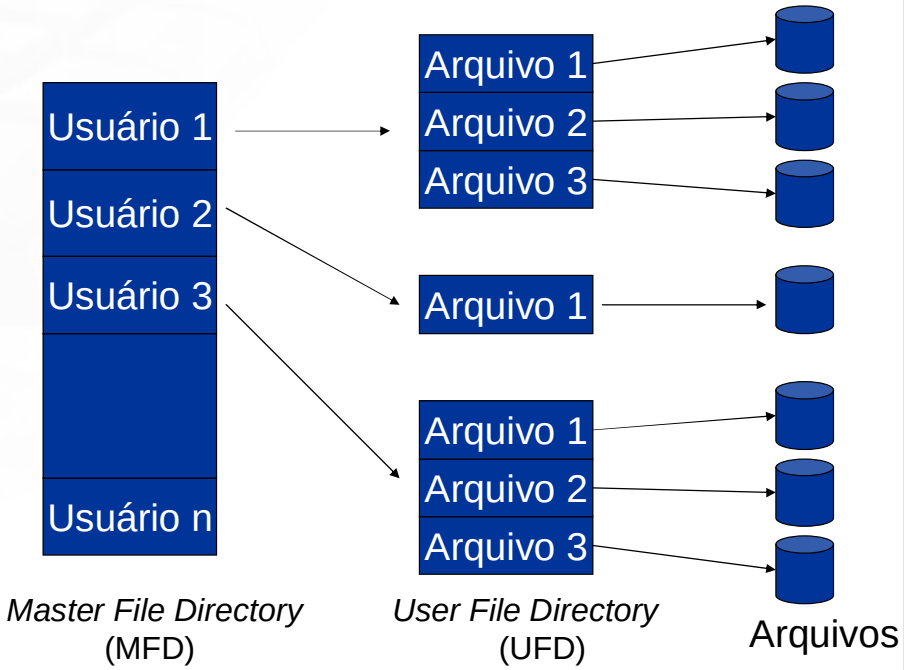
- Modo como o sistema organiza os diferentes arquivos contidos num disco
- É a estrutura de dados que contém entradas associadas aos arquivos
  - localização física, nome, organização e demais atributos
- Quando um arquivo é aberto, o sistema operacional procura a sua entrada na estrutura de diretórios
- As informações do arquivo são armazenadas em uma tabela mantida na memória principal(tabela de arquivo abertos)
  - Fundamental para aumentar o desempenho das operações com arquivos

Diretórios (2)

- Sistemas de Diretório em Nível Único
  - Implementação mais simples
  - Existe apenas um único diretório contendo todos os arquivos do disco
  - Bastante limitado já que não permite que usuários criem arquivos com o mesmo nome
    - Isso ocasionaria um conflito no acesso aos arquivos

Diretórios (3)

- Estrutura de diretórios com dois níveis
  - Para cada usuário existe um diretório particular e assim poderia criar arquivos com qualquer nome.
  - Deve haver um nível de diretório adicional para controle que é indexado pelo nome do usuário
    - Cada entrada aponta para o diretório pessoal.

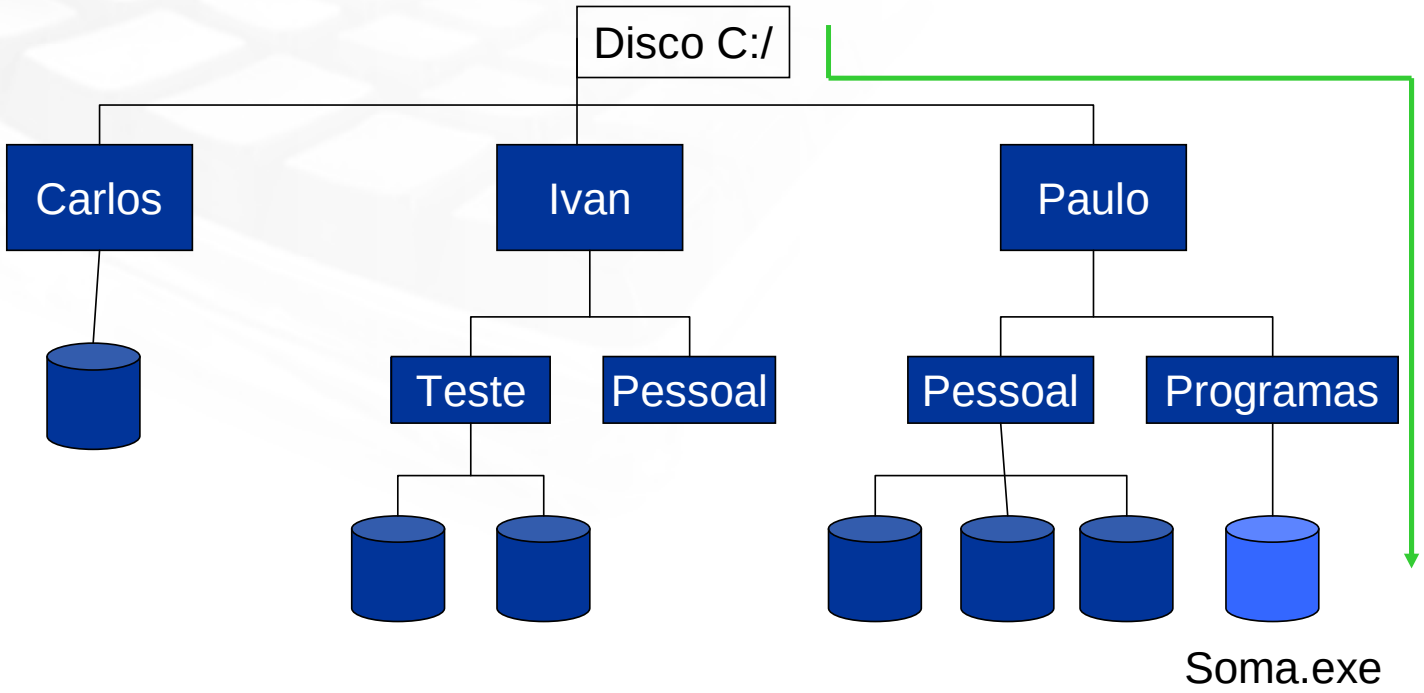


Diretórios (4)

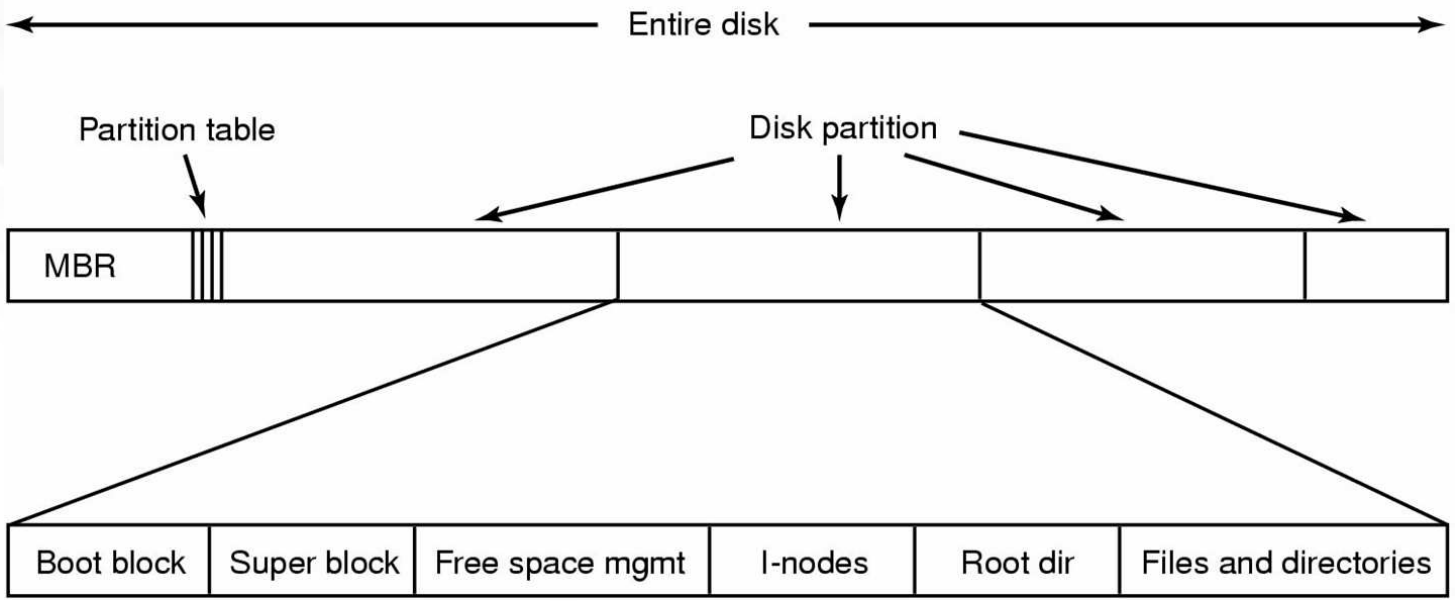
- Estrutura de diretórios Hierárquicos
  - Adotado pela maioria dos sistemas operacionais
  - Logicamente melhor organizado
  - É possível criar quantos diretórios quiser
  - Um diretório pode conter arquivos e outros diretórios (chamados subdiretórios)
  - Cada arquivo possui um **path** único que descreve todos os diretórios da raiz (MFD) até o diretório onde o arquivo esta ligado
  - Na maioria dos S.O.s os diretórios são tratados como arquivos tendo atributos e identificação

Diretórios (4)

- Estrutura de diretórios Hierárquicos (cont.)



Esquema do Sistema de Arquivos (1)



Esquema do Sistema de Arquivos (2)

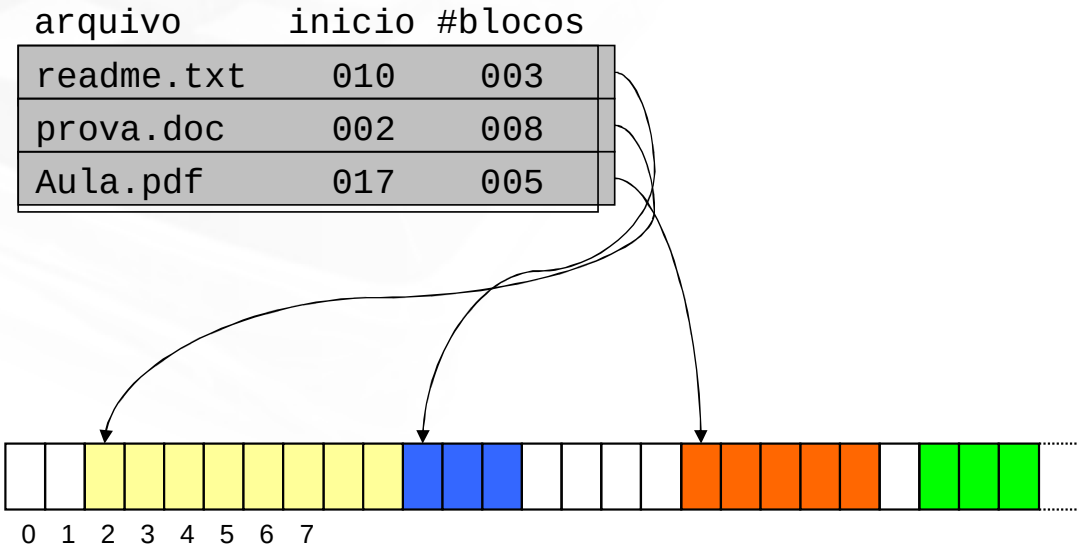
- A maioria dos discos é dividida em uma ou mais partições com Sistemas de arquivos independentes para cada partição
- O setor 0 do disco é chamado de *Master Boot Record* (MBR)
- Na inicialização do sistema, a BIOS lê e executa o MBR
  - O programa do MBR localiza a partição ativa, lê seu primeiro bloco, chamado de **bloco de boot (boot sector no NTFS)**
  - O programa no bloco de boot carrega o S.O. contido na partição
- O esquema da partição varia de um S.O. para outro, mas é comum:
  - A definição de um **SuperBloco (Master File Table no NTFS)**: contém os principais parâmetros do sistema de arquivos (tipo, no. de blocos, etc.)
  - As informações sobre os blocos livres
  - FCB (File control block) define detalhes do arquivo, incluindo permissões, propriedade, tamanho, e local dos blocos de dados. No UNIX (Inode), NTFS (fica no MFT que usa estrutura de banco de dados relacional, uma linha por arquivo)

Implementação de Arquivos (1)

- Alocação Contígua
  - Consiste em armazenar um arquivo em blocos seqüencialmente dispostos
  - O sistema localiza um arquivo através do endereço do primeiro bloco e da sua extensão em blocos
  - O acesso é bastante simples
  - Seu principal problema é a alocação de novos arquivos nos espaços livres
    - Para armazenar um arquivo que ocupa **n** blocos, é necessário uma cadeia com **n** blocos dispostos seqüencialmente no disco
  - Além disso, como determinar o espaço necessário a um arquivo que possa se estender depois da sua criação?
    - Pré-alocação (fragmentação interna)

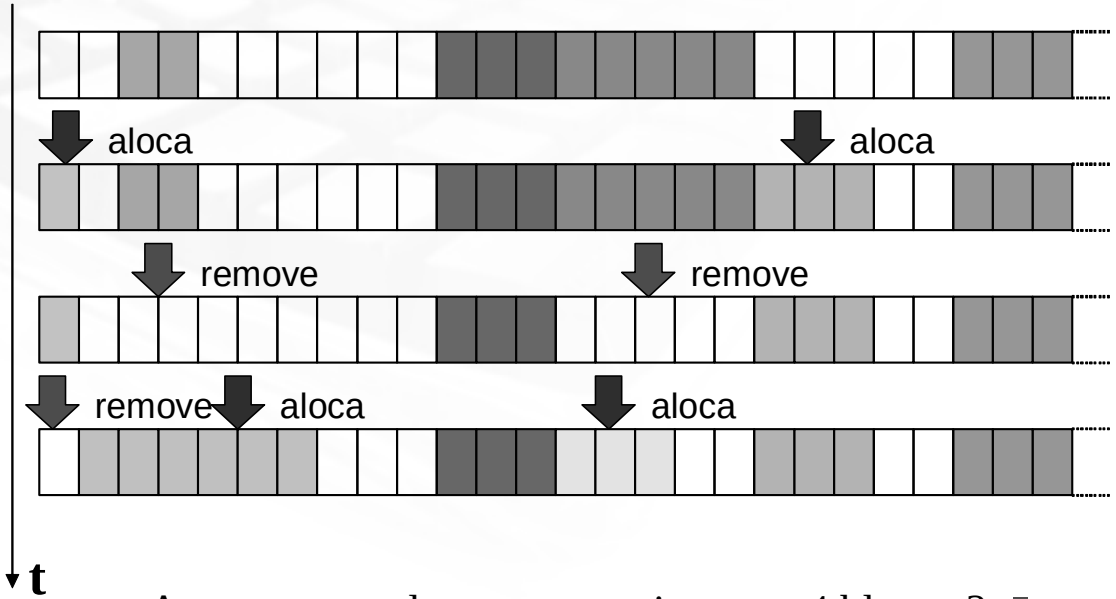
Implementação de Arquivos (2)

- Alocação Contígua (cont.)



Implementação de Arquivos (3)

- Alocação Contígua (cont.)



Agora, como alocar um arquivo com 4 blocos ? Fragmentação Externa !

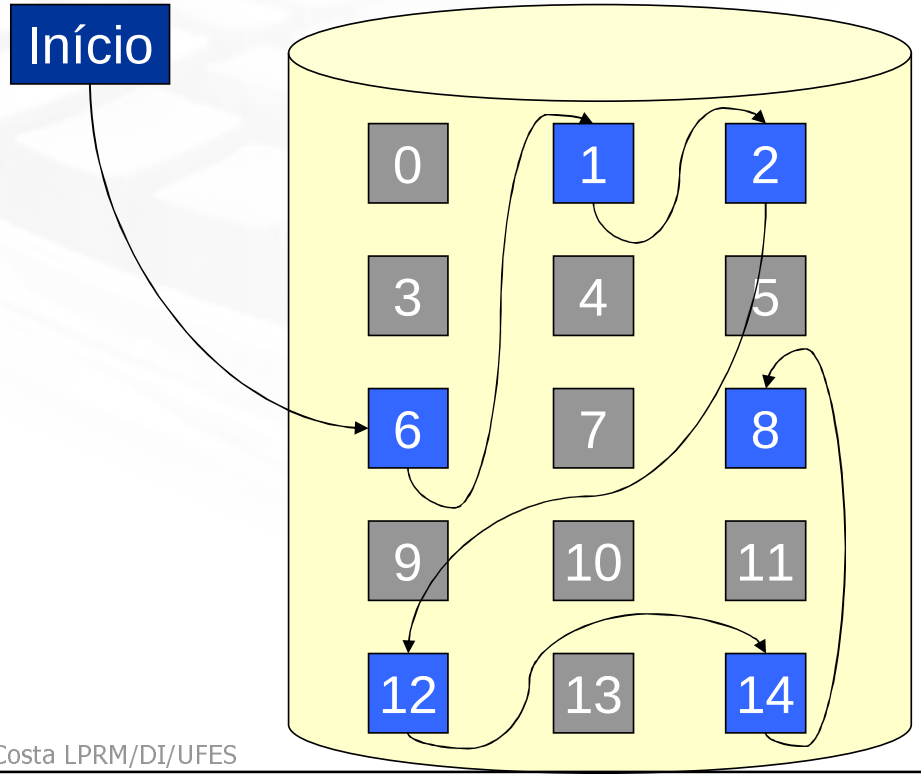
E se o arquivo fosse dividido em Blocos lógicos?

Implementação de Arquivos (4)

- Alocação por Lista Encadeada
  - O arquivo é organizado como um conjunto de blocos ligados no disco
  - Cada bloco deve possuir um ponteiro para o bloco seguinte
  - Aumenta o tempo de acesso ao arquivo, pois o disco deve deslocar-se diversas vezes para acessar todos os blocos
  - É necessário que o disco seja desfragmentado periodicamente
  - Esta alocação só permite acesso seqüencial
  - Desperdício de espaço nos blocos com armazenamento de ponteiros

Implementação de Arquivos (5)

- Alocação por Lista Encadeada (cont.)



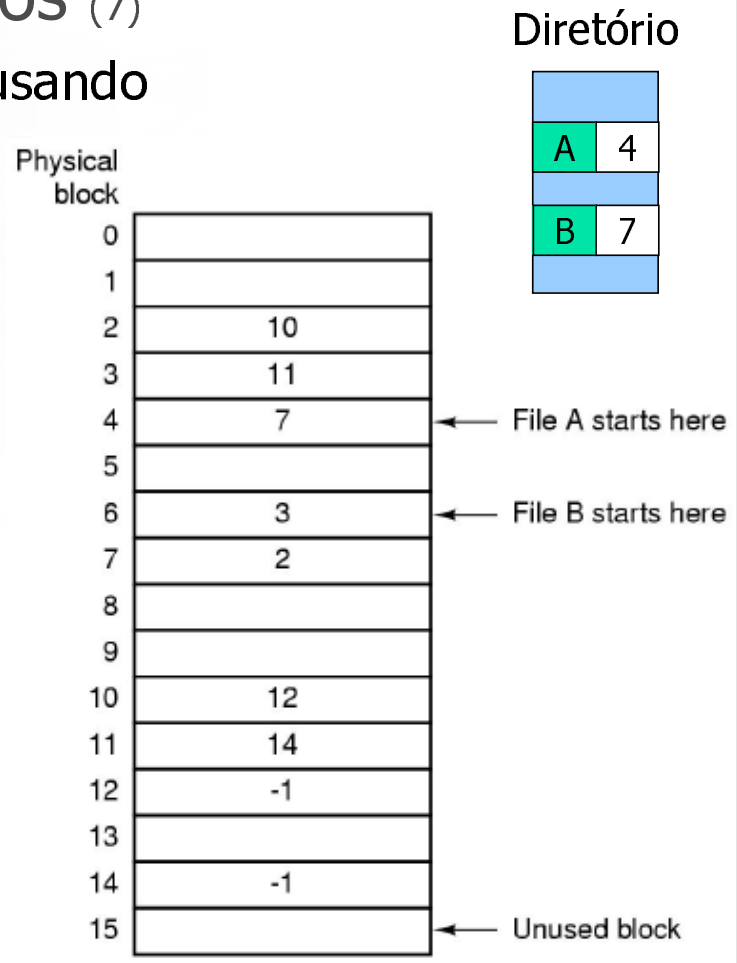
Implementação de Arquivos (6)

- Alocação por Lista Encadeada usando Tabela na Memória
  - Mantém os ponteiros de todos os blocos de arquivos em uma única estrutura denominada **Tabela de Alocação de Arquivos**
    - **FAT** (File Allocation Table)
  - Vantagens:
    - Permitir o acesso direto aos blocos
    - Não mantém informações de controle dentro dos blocos de dados
  - Esquema usado pelo MS-DOS, Win95 e Win98

Implementação de Arquivos (7)

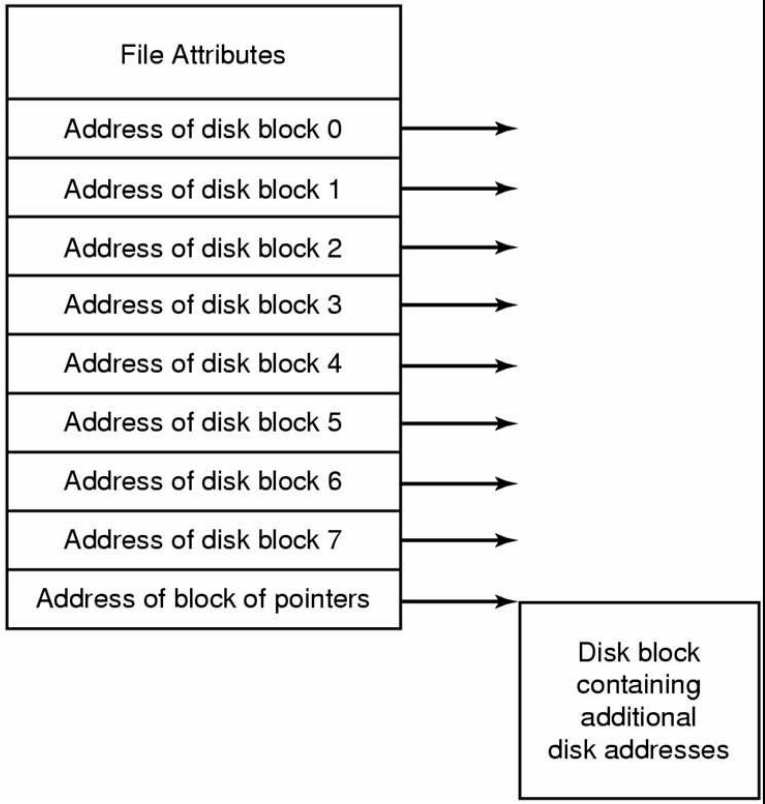
- Alocação por Lista Encadeada usando Tabela na Memória (cont.)

- Desvantagem
  - A tabela deve estar na memória
  - Disco de 20 G, blocos de 1k?
    - 20 milhões de entradas
    - Cada entrada 4 bytes: 80MB
  - Mas é possível paginar a FAT!



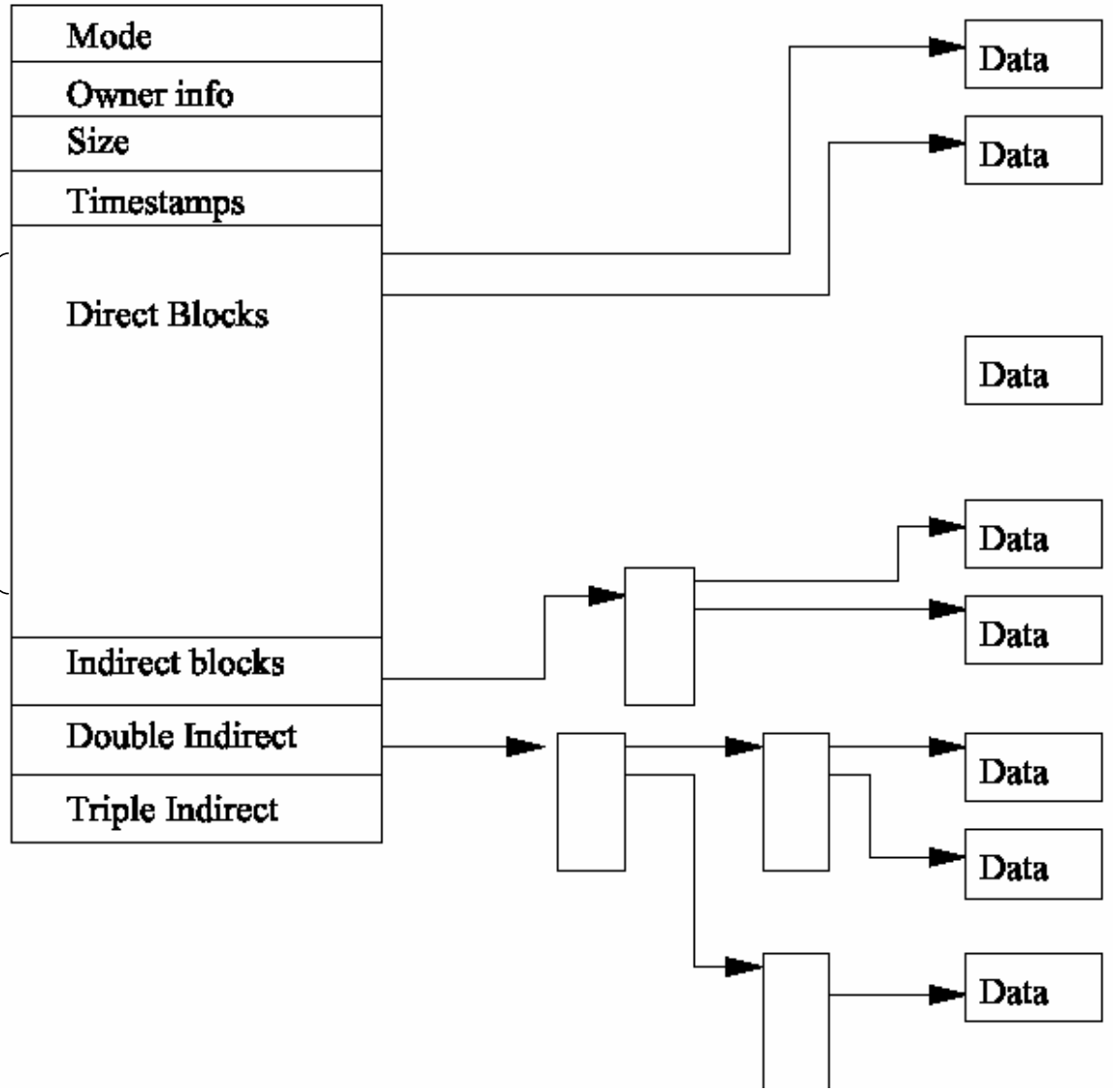
Implementação de Arquivos (8)

- i-nodes
  - Cada arquivo possui uma tabela (i-node) no disco
  - O i-node só precisa estar na memória quando o arquivo correspondente estiver aberto
  - Ocupa menos espaço que a FAT
    - Tamanho da FAT cresce linearmente com o tamanho do disco
    - I-nodes requerem um espaço proporcional à quantidade máxima de arquivos abertos
  - Usados por sistemas baseados no UNIX



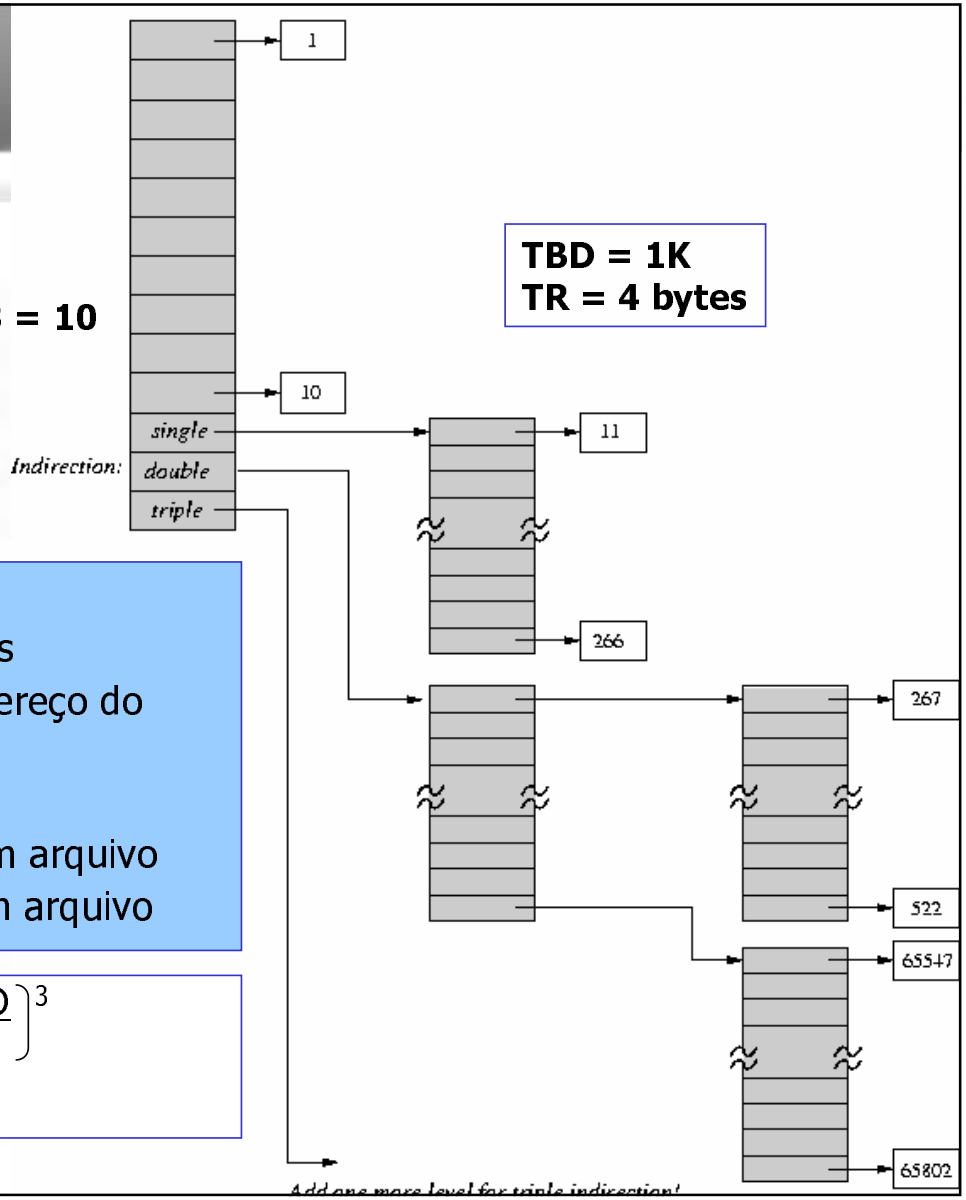
i-nodes (1)

10 entradas diretas



i-nodes (2)

Bmax=10 + 256 + 2562 + 2563 = 10 + 216 + 224 = 10 + 64K + 16M blocos
Fmax » 16 Gigabytes



TBD = 1K
TR = 4 bytes

Dado que:
- TBD – Tamanho bloco de dados
- TR – Tamanho referência (endereço do bloco)

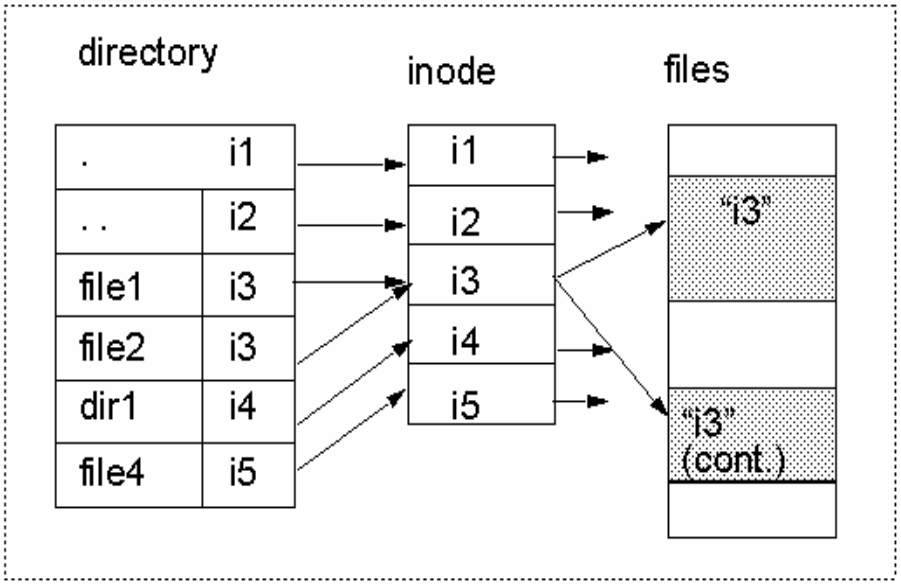
Qual será...?
- B_{max} – Nº Blocos máximo de um arquivo
- F_{max} – dimensão máxima de um arquivo

$$B_{max} = 10 + \frac{TBD}{TR} + \left(\frac{TBD}{TR}\right)^2 + \left(\frac{TBD}{TR}\right)^3$$

$$F_{max} = B_{max} \times TBD$$

Relação entre Diretórios e i-nodes (1)

- Diretórios incluem nomes de arquivos e referências para os respectivos i-nodes



Relação entre Diretórios e i-nodes (2)

- Passos para alcançar /usr/ast/mbox