Sistemas Operacionais

Implementação do Sistema de Arquivos

Lesandro Ponciano

2025

Objetivos da Aula

- Descrever alguns detalhes de implementação de estruturas de sistemas de arquivos

- Discutir estratégias de
  - alocação de blocos
  - gerenciamento do espaço livre

Sistemas de Arquivos

- Sistema de arquivos composto de níveis
  - Controle de I/O: é composto por *drivers* de dispositivos
  - Sistema de arquivos básico: controla caches do sistema de arquivos e *buffers*
  - Módulo de organização de arquivo: converte endereços de bloco lógicos em endereços de blocos físicos
  - O nível de sistema de arquivos lógico controla informações de metadados por meio do bloco de controle de arquivos (FCB)

Acesso aos Dados

- O sistema de arquivos mantém algoritmos e estruturas de dados que mapeiam o sistema de arquivos lógico para os dispositivos físicos de memória secundária

- Para prover a eficiência de I/O, as transferências de I/O entre a memória principal e o disco são executadas em unidades de bloco
  - Cada bloco tem um ou mais setores

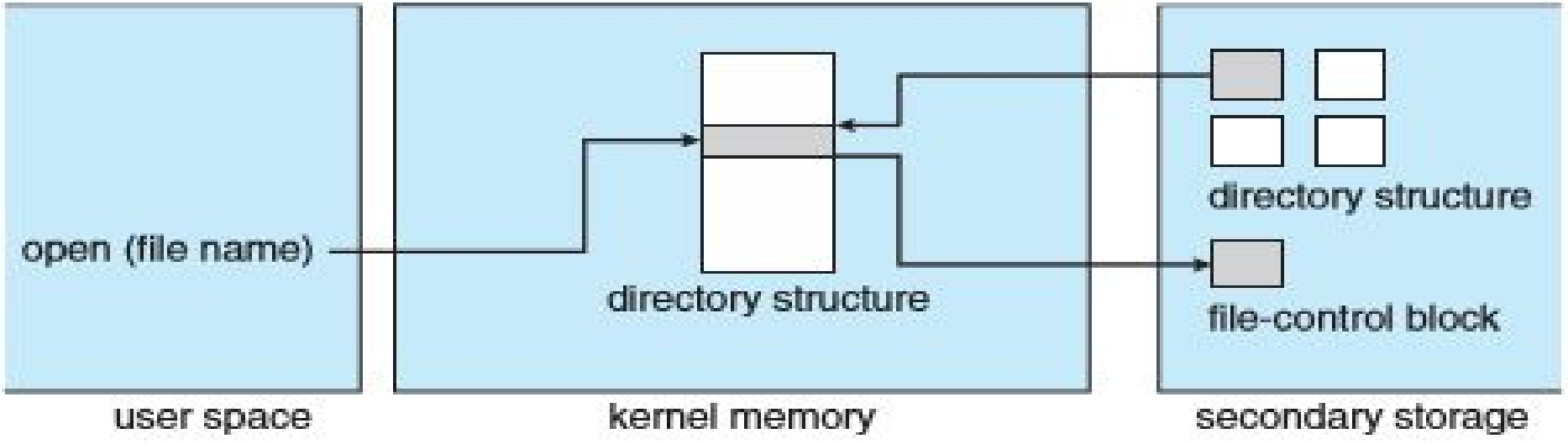
Informações Mantidas

- Várias estruturas de disco e de memória são utilizadas pelo sistema de arquivos

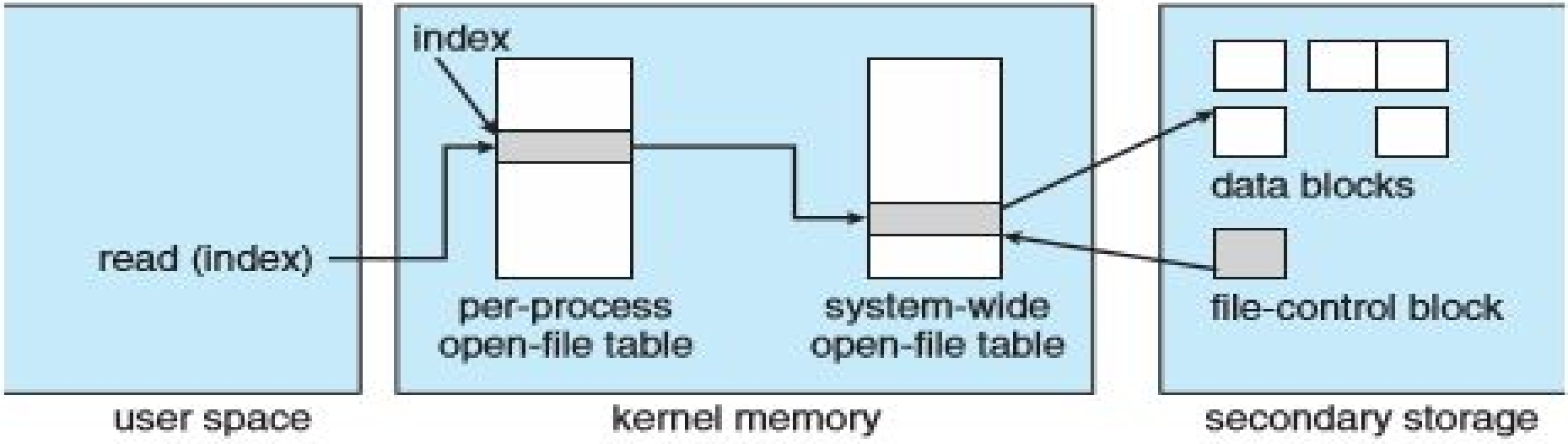
- Informações mantidas em disco
  - Bloco de controle de inicialização; Bloco de controle de volume; Estrutura de diretórios; *File control block*- FCB

- Informações mantidas em memória
  - Tabela de montagens; Cache de estrutura de diretórios; Tabela de arquivos abertos por processo; *Buffers*

Estruturas na Abertura de Arquivo



Estruturas na Leitura de Arquivo



Inicialização

- Informações de inicialização
  - Têm um formato próprio, pois ainda não há sistemas de arquivos carregados
  - Geralmente são uma série de blocos carregados como imagem na memória
  - A execução da imagem começa em uma locação predefinida

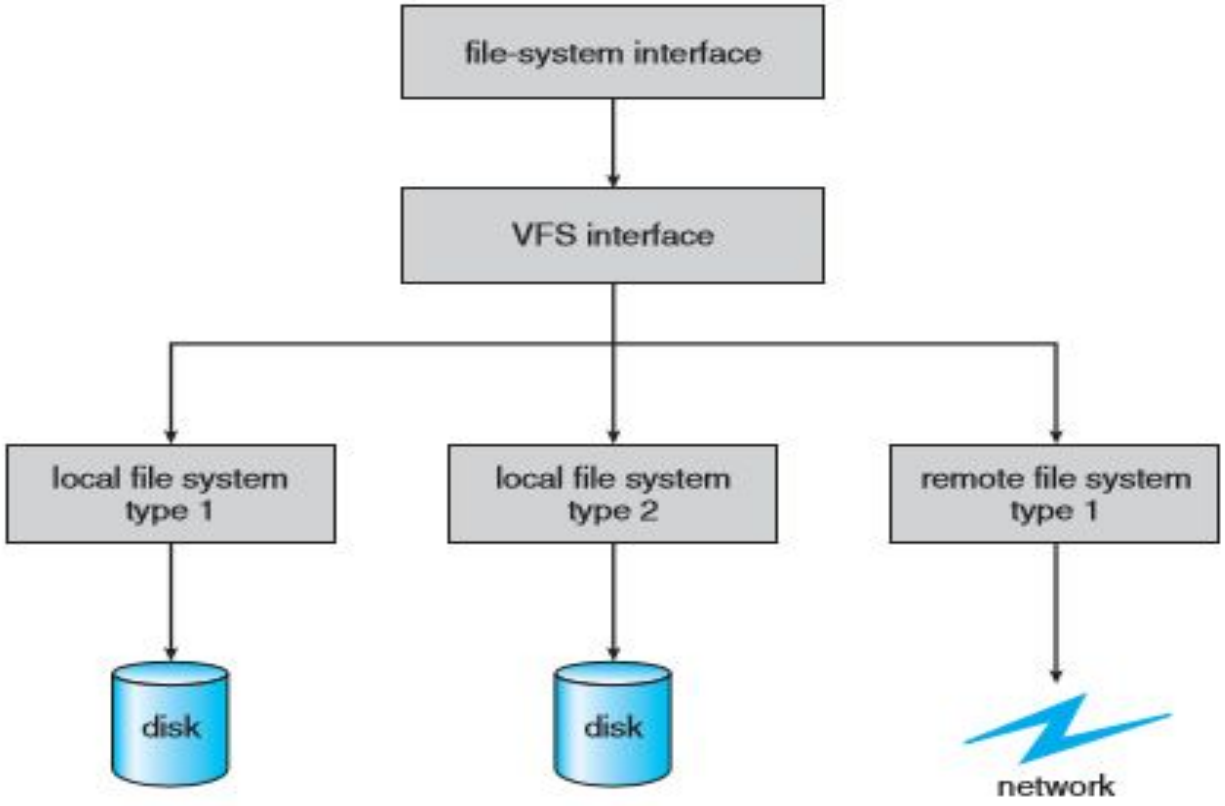
- Carregador de inicialização
  - Conhece suficientemente a estrutura do sistema de arquivos para encontrar e carregar o *kernel*
  - Pode conhecer diversos sistemas de arquivos (exemplo: inicialização dual)

Sistema de Arquivos Virtuais

- Um SO pode conhecer diversos sistemas de arquivos
  - Isso é feito por meio de uma interface de sistema de arquivos virtual, VFS (*virtual file system*)

- Funções do VFS
  - Separar operações genéricas do sistema de arquivos de sua implementação
  - Fornecer um mecanismo para a representação exclusiva de um arquivo em toda a rede

Esquema de VFS



Implementação

- Três problemas de implementação são especialmente relevantes
  1) Implementação do diretório
  2) Alocação de espaço em disco
  3) Gerenciamento de espaços livres

1 Implementação do Diretório

- Lista linear
  - Lista linear de nomes de arquivos com ponteiros para os blocos de dados
  - Para se criar um novo arquivo, é necessário percorrer toda a lista para identificar se já não há outro arquivo com mesmo nome
- Tabela com *Hash*
  - Lista linear armazena as entradas do diretório
  - Tabela com *hash* recebe um valor calculado a partir do nome do arquivo e retorna o ponteiro para o nome do arquivo na lista linear
  - É rápido descobrir se já existe um arquivo com um dado nome

2 Métodos de Alocação

- Como alocar espaço em disco de forma que os arquivos sejam armazenados de forma eficiente e que permita acesso rápido?
  - Alocar blocos livres suficientes para armazenar o arquivo
  - Blocos lógicos do disco são numerados sequencialmente

- Três formas
  - Alocação Contígua
  - Alocação Encadeada
  - Alocação Indexada

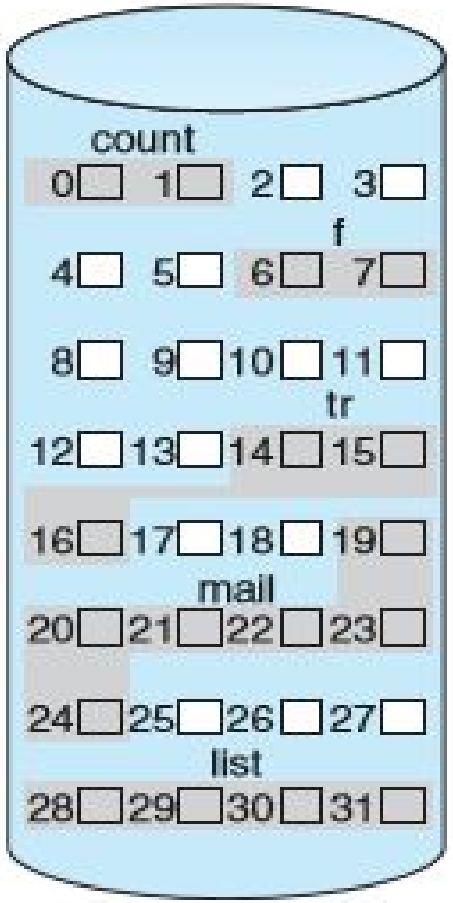
Alocação Contígua

- Arquivo é uma sequência de blocos lógicos contíguos alocados no momento da criação

- Endereços no disco são lineares
  - Bloco lógico *i* e *i+1* são armazenados fisicamente em sequência
  - Reduz a necessidade de procura (*seek*), já que blocos estão na mesma linha

- Arquivo é descrito através de uma entrada com
  - Bloco físico inicial
  - Tamanho do arquivo em blocos

Alocação Contígua



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Problemas

- Encontrar espaço para um novo arquivo
  - Técnicas de gerência (semelhante às usadas na memória principal), ex: *first fit*, *best fit*, *worst fit*
  - Gera fragmentação externa no disco, necessidade de compactação

- Determinar o espaço necessário para um arquivo
  - Arquivos tendem a crescer, e se não há espaço contíguo disponível?
    - Abortar a execução do programa com um erro
    - Pré-alocar um espaço máximo para o arquivo

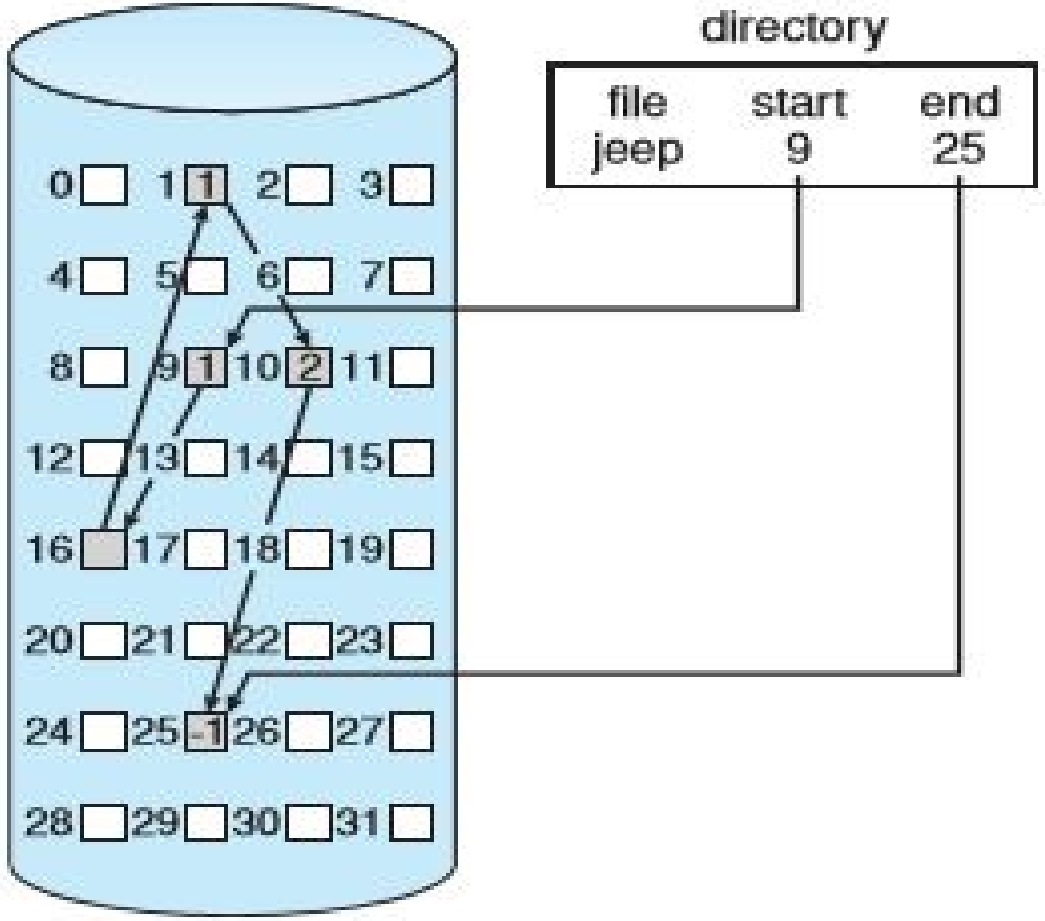
Alocação em Lista Encadeada

- Arquivo é uma lista encadeada de blocos, na qual cada bloco contém um ponteiro para o próximo

- Arquivo é descrito com
  - Bloco inicial
  - Bloco final ou tamanho do arquivo em blocos

- Soluciona os problemas da alocação contígua em relação ao dimensionamento do tamanho e crescimento de arquivos

Lista Encadeada



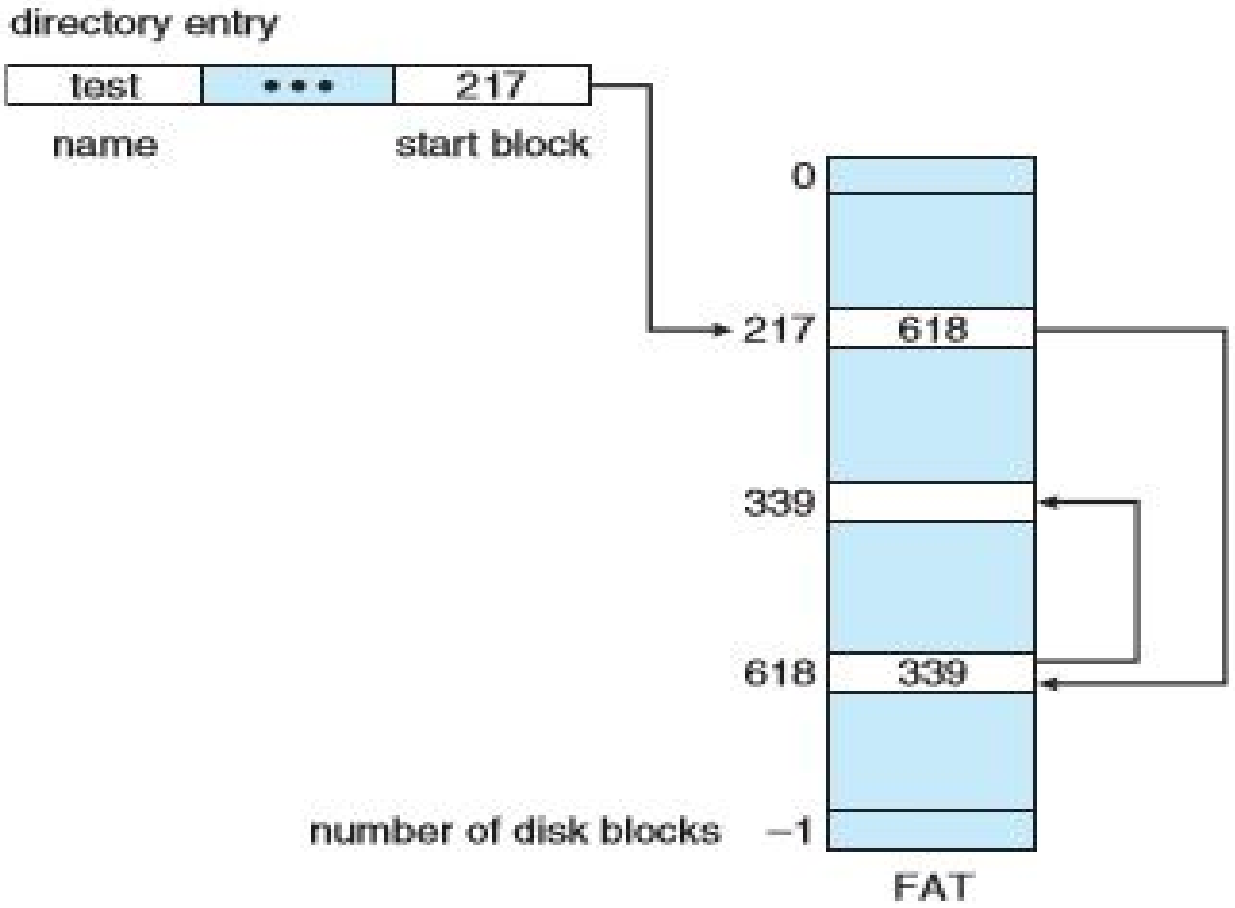
Lista Encadeada

- Elimina a fragmentação externa e arquivos podem crescer indefinidamente
  - Não há necessidade de compactar o disco
- O acesso a um bloco *i* implica em percorrer a lista encadeada
  - Afeta o desempenho
  - Adequado para acesso sequencial a arquivos
- Confiabilidade: erro provoca a leitura/escrita em bloco pertencente a outro arquivo

Tabela de Alocação de Arquivos (FAT)

- *File Allocation Table* (FAT)
- Variação de alocação encadeada
- FAT é uma tabela de encadeamento de blocos lógicos
  - Há uma entrada na FAT para cada bloco lógico do disco (sistema de arquivos)
  - Composta por um ponteiro (endereço do bloco lógico)
  - Arquivo é descrito por uma sequência de entradas na FAT, cada entrada apontando para a próxima entrada

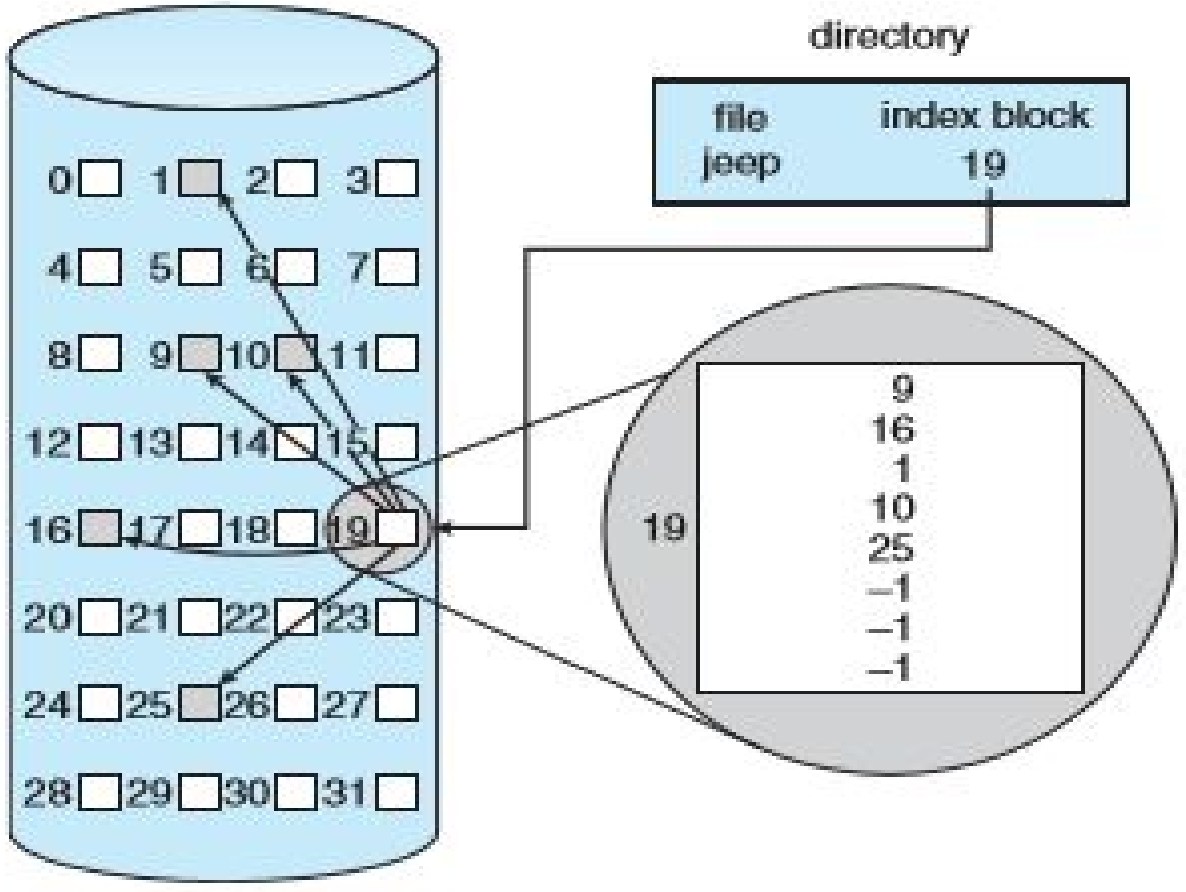
FAT


directory entry
test
• • •
217
name
start block
0
217
618
339
339
618
number of disk blocks
−1
FAT

Alocação Indexada

- Busca resolver o problema da grande quantidade de ponteiros pelo disco, como ocorre na alocação encadeada
- Mantém, por arquivo, um índice de blocos que pertencem a ele
- O índice é mantido em um bloco
- Diretório possui um ponteiro para o bloco onde está o índice associado a um determinado arquivo

Alocação Indexada



Alocação Indexada

- Permite o acesso randômico a blocos, independente de sua posição relativa no arquivo
- Tamanho máximo do arquivo é limitado pela quantidade de entradas suportadas pelo bloco
  - Poucas entradas limita o tamanho do arquivo
  - Muitas entradas desperdiça espaço no disco
- Solução é utilizar dois tamanhos de blocos, um para índice e outro para dados

Atividade de Fixação 1

- Para que serve um sistema de arquivos virtual?

- Diferencie, por meio de desenhos, as alocações contígua, encadeada e indexada. Destaque um problema em cada uma delas.

3 Gerenciamento do Espaço Livre

- Quando possível, é importante reutilizar o espaço que era ocupado por arquivos que foram excluídos
- Para controlar o espaço livre em disco, o sistema operacional mantém uma lista de espaços livres
  - Registra todos os blocos do disco que não estão alocados para algum arquivo ou diretório
- Ao ser criado, um arquivo usa espaços que estão na lista de espaços livres e, ao ser removido, os espaços são adicionados novamente em tal lista

Implementação

- A lista de espaços livres pode ser implementada como
  - Vetor de bits
  - Lista encadeada
  - Agrupamento
  - Contagem

Implementação em Vetor de Bits

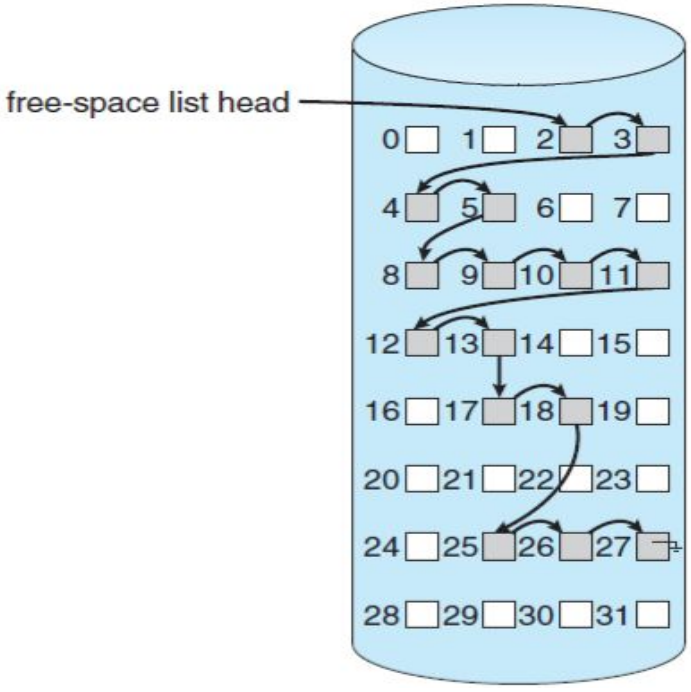
- Também chamado de Mapa de Bits
- Cada bloco é representado por um bit
  - Se o bloco está livre, o bit é 1
  - Se o bloco está alocado, o bit é 0

Estão livres os blocos: 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, 27

- O mapa ocupa espaço no disco:
  - Um disco de 1 TB com blocos de 4 KB requer 32 MB para armazenar o mapa de bits

Implementação em Lista Encadeada

- Encadeamento de todos os blocos livres no disco
- Há um ponteiro para o primeiro bloco livre em uma locação especial no disco
- Para percorrer a lista, é necessário ler cada bloco. Isso não é eficiente, pois requer um tempo de I/O significativo



- Estão livres os blocos: 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, 27

Implementação como Agrupamento

- Agrupa-se os endereços de diversos blocos livres em um bloco livre
- Armazena-se o endereço de *n* blocos livres no primeiro bloco livre
  - Os *n-1* blocos livres ficam realmente livres
- Permite encontrar o endereço de diversos blocos livres rapidamente

Implementação em Contagem

- Estratégia usada em alocação contígua
- Em vez de manter uma lista com *n* endereços livres, mantém-se o endereço do primeiro bloco livre e a quantidade (*n*) de blocos contíguos livres que estão após o primeiro
- Pode ser implementado usando Árvore B, de forma que a inserção, exclusão e pesquisa sejam eficientes

Eficiência e Desempenho

- Dos principais componentes do computador, o disco é o mais lento
  - Por isso, ele é uma das principais causas de gargalo

- Há diversas técnicas que são usadas para tentar aumentar o desempenho do sistema de arquivos
  - Uso de cache *buffer* unificada
  - Uso adequado de gravações síncronas (não usam o cache) e assíncronas (usam o cache)
  - Para liberar espaço no *buffer*, uso de técnicas *free-behind* e *read-ahead*

Recuperação

- Mecanismos de consistência e recuperação de dados no disco geralmente se baseiam em:
  - Identificar e resolver inconsistências entre as estruturas de dados e os dados (blocos) no disco
    - Unix (*fsck*) e Windows (*chkdsk*)
  - Usar informações de *logs* de operações no sistema de arquivos para identificar inconsistências e tratá-las
  - Uso de backups

Atividade de Fixação 2

- Diferencie as seguintes técnicas de implementação da lista de espaços livres:
  - Vetor de bits
  - Lista encadeada
  - Agrupamento
  - Contagem

Referências

TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. São Paulo: Pearson Prentice Hall, 2009. xvi, 653 p. ISBN 9788576052371

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais: princípios básicos. Rio de Janeiro, RJ: LTC, 2013. xvi, 432 p. ISBN 9788521622055

Sistemas Operacionais: Sistemas de Arquivos. http://www.inf.ufrgs.br/~asc/livro/transparencias/cap8.pdf

Sistemas Operacionais

Prof. Dr. Lesandro Ponciano

https://orcid.org/0000-0002-5724-0094