GSI018 – Sistemas Operacionais

Implementação de Sistemas de Arquivos

Universidade Federal de Uberlândia
Faculdade de Computação
Prof. Dr. rer. nat. Daniel D. Abdala

Na Aula Anterior...

- Memória vs Armazenamento;
- Sistema de Arquivos;
- Revisão Sobre Dispositivos de Armazenamento;
- A Abstração – Arquivo;
- Nomes;
- Extensões;
- Formatos de Arquivos;
- Estruturas de Arquivos;
- Tipos de Arquivos;
- Atributos de Arquivos;
- Formas de Acessar Arquivos;
- Operações com Arquivos;
- Diretórios;
- Caminhos;
- Operações com Diretórios;

Nesta Aula

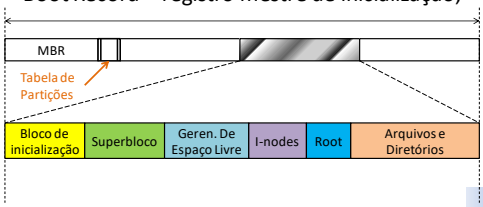
- Esquema do Sistema de Arquivos;
- Alocação Contígua;
- Alocação por Lista Encadeada;
- Alocação por Lista Encadeada Usando uma Tabela na Memória;
- I-nodes;
- Implementação de Diretórios;
- Sistemas de Arquivos Virtuais;

Esquema do Sistema de Arquivos

- Ponto de Vista do Sistema;
- Aspectos de Implementação;

Estrutura de Baixo Nível de um Disco

- A maioria dos discos é organizado tal como na figura abaixo;
- O setor 0 do disco é chamado de MBR – Master Boot Record – registro mestre de inicialização;



Estrutura de Baixo Nível de um Disco

- O MBR é utilizado para inicializar o computador;
- O fim do MBR contém a Tabela de Partições, que indica os endereços iniciais e finais de cada partição;
- Uma das partições na tabela é marcada como ativa;
- Quando o computador é inicializado, a BIOS lê e executa o MBR:
  - Localizar a partição ativa;
  - Ler o 1º bloco da partição ativa (bloco de inicialização)

Estrutura de Baixo Nível de um Disco

- **Superbloco** → Contém todos os principais parâmetros sobre o sistema de arquivos tal como tamanho do bloco, nº de blocos, nº mágico, etc;
  - Lido do disco e salvo na memória quando o computador é inicializado ou quando o sistema de arquivos é utilizado pela primeira vez;
  - Nº mágico – indica o tipo de sistema de arquivos;
- **Gerenciamento de Espaço Livre** → implementado via mapas de bits, listas livres, etc...
- **I-nodes** → blocos de indexação;
- **Root** → diretório base da árvore de diretórios;
- **Arquivos e Diretórios** → blocos propriamente ditos;

Implementação do Sistema de Arquivos

- Controle de quais blocos de disco estão relacionados a quais arquivos;
- Projeto depende:
  - Do dispositivo;
  - Do sistema de arquivos (FAT32, NTFS, EXT2, EXT3, EXT4, etc);

Alocação Contígua

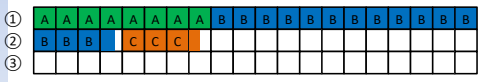
- Esquema mais simples;
- Arquivo é armazenado em blocos contíguos no disco;
- Arquivos são sempre armazenados iniciando no início do bloco;
- Caso o tamanho do arquivo não coincida com um múltiplo do tamanho de um bloco, espaço em disco será desperdiçado;

Alocação Contígua

- Vantagens:
  - Simples de implementar – dois números, bloco inicial e final;
  - Desempenho de leitura é excelente, pois todo o arquivo encontra-se armazenado sequencialmente no disco;
- Desvantagens:
  - Com o tempo, o disco fica fragmentado
- Hoje em dia é adequado apenas para sistemas de arquivos nos quais sabe a priori o tamanho dos arquivos (ex: CDs)

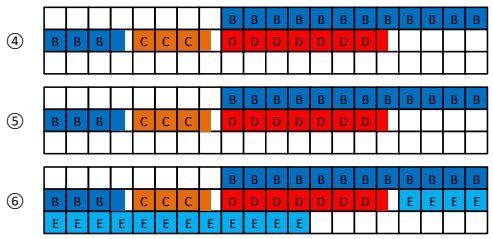
Exemplo Alocação Contígua

1. **(INS) Arquivo A = 32KB;**
2. **(INS) Arquivo B = 63KB;**
3. **(INS) Arquivo C = 14KB;**
4. (DEL) Arquivo A;
5. **(INS) Arquivo D = 30KB;**
6. (DEL) Arquivo C;
7. **(INS) Arquivo E = 64KB;**
8. **(INS) Arquivo F = 32KB;**
9. (DEL) Arquivo D;
10. **(INS) Arquivo G = 64KB;**

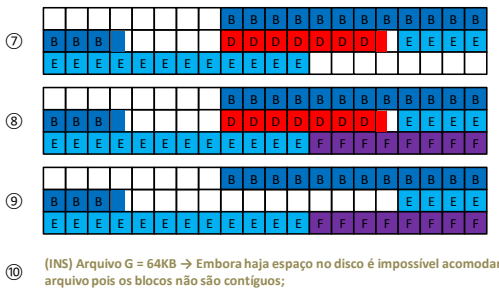


→tamanho do bloco = 4KB

Exemplo Alocação Contígua



Exemplo Alocação Contígua

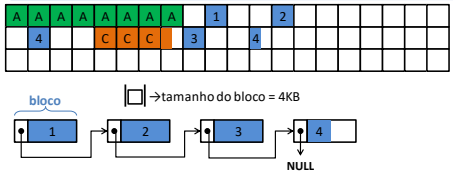


(INS) Arquivo G = 64KB → Embora haja espaço no disco é impossível acomodar o arquivo pois os blocos não são contíguos;

Alocação por Lista Encadeada

- Lista encadeada de blocos em disco;
- A primeira palavra de cada bloco é um ponteiro para o próximo bloco usado pelo arquivo;
- Qualquer bloco pode ser usado, consequentemente nenhum espaço em disco é disperdiçado pela fragmentação;
- Indexar é mais simples – basta registrar qual o primeiro bloco do arquivo;
- Acesso aleatório é extremamente lento!
  - Para acessar o bloco n (note, queremos apenas o bloco n) todos os n-1 blocos devem ser lidos;
- Quantidade de bytes não é mais uma potência de 2!

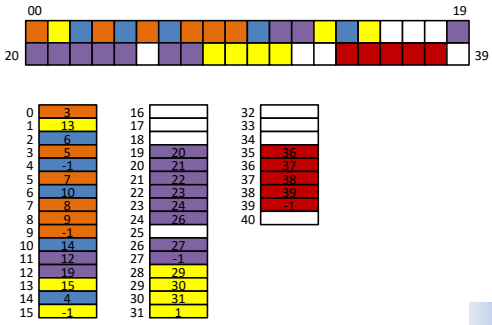
Alocação por Lista Encadeada



Alocação por Lista Encadeada Usando uma Tabela na Memória

- Busca eliminar as desvantagens da alocação por lista encadeada;
- Coloca as palavras do ponteiro de cada bloco em uma tabela na memória;
- Esta tabela na memória principal é chamada de **FAT – File Allocation Table**;
- Mas rápido acessar dados diretamente da memória;
- Tabela inteira deve existir na memória;

Alocação por Lista Encadeada Usando uma Tabela na Memória



Exemplo

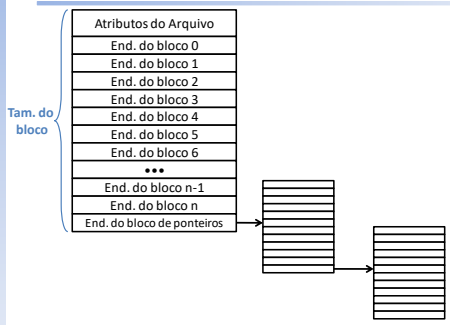
- HD – 200GB – 200 x 10^9;
- Bloco – 1KB – 1x10^3;
- 200x10^6 blocos;
- Pelo menos 28 bits para endereços → 32 bits para endereços;
- 4bytes x 200M blocos = 800MB de espaço em memória;

I-nodes

- Um tipo de estrutura de dados;
- Utilizada na maioria dos sistemas de arquivos modernos;
- É necessário apenas um ponteiro para o primeiro i-node do arquivo para indexá-lo;
- Os requisitos de memória são muito menores se comparados com a alocação por lista encadeada usando uma tabela na memória;
- A tabela precisa ser de no máximo NxK bits, onde N é o tamanho do i-node e K é o número máximo de arquivos que podem estar abertos no sistema em um dado momento;

19

I-nodes



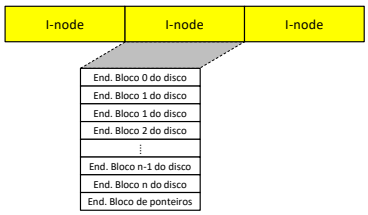
20

I-nodes

- Considere um sistema de arquivos que utiliza blocos de 2K bytes;
- Um HD de 1TB;
- $10^{12}/2x10^3=½x10^9=500$ milhões de blocos
- Endereços de 32 bits são suficientes;
- Considerando que os atributos ocupem 256 bytes sobram 1.792 bytes para endereços de blocos (448);
- Utilizamos o último endereço para um ponteiro para blocos de endereços;
- 1 i-node é capaz de representar um arquivo de 447x2.048= 915.456 bytes;

21

Estrutura do i-node no Disco



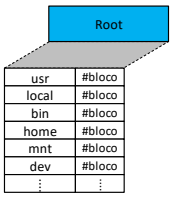
22

Implementação de Diretórios

- "/" (root) é um diretório especial;
  - Sempre aberto;
  - Alocação direta no início do disco;
- Todos os demais diretórios são tratados como um tipo especial de arquivo;
- Diretórios possuem um nome e atributos associados;
- Como representar em baixo nível um diretório depende do sistema de arquivo;

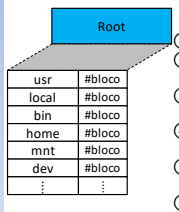
23

Exemplo de Diretório ROOT



24

Abertura de um Diretório



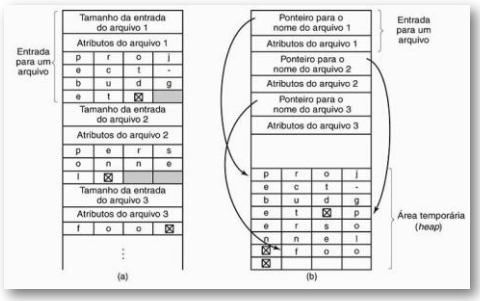
ls /home

1. O programa ls é executado pelo shell;
2. O programa executa a chamada do sistema OPENDIR("/home");
3. Sistema de arquivos resolve o caminho e encontra o diretório "home" na raiz;
4. O nº do 1º bloco associado a "home" é recuperado da raiz e carregado como um i-node;
5. O 1º bloco de dados indexado pelo i-node é carregado na memória;
6. O programa ls lê então o conteúdo do arquivo utilizando a chamada do sistema READDIR;
7. Eventualmente o i-node é consultado para buscar mais blocos de dados do disco;
8. Ao final da leitura do diretório a chamada do sistema CLOSEDIR é invocada fechando o diretório;
9. O i-node associado ao diretório home é liberado;

Implementação de Diretórios

- Decisões de Implementação:
  - Onde armazenar os nomes?
  - Onde armazenar os atributos?
  - Número máximo de arquivos?
- DOS/Windows – nomes de tamanho máximo fixo;
- LINUX – nomes de tamanho variável;

Formas de Codificação das Entradas em um Diretório



Tanenbaum, A. S. Sistemas Operacionais Modernos, 3ª Edição. Pearson.

Sistemas de Arquivos Virtuais

- Diferentes sistemas de arquivos em uso no mesmo sistema computacional;

Ex: Um Sistema Computacional Windows

- Principal – **NTFS**
- CD-ROM – **ISO 9660**
- HD-Antigo – **FAT-16**
- DVD – **UDF**

- Cada sistema de arquivos distinto é identificado por uma unidade (C:, D:, ...)
- Usando a unidade, o sistema operacional sabe para qual sistema de arquivos repassar a requisição;
- O windows não tenta unificar sistemas de arquivos heterogêneos;

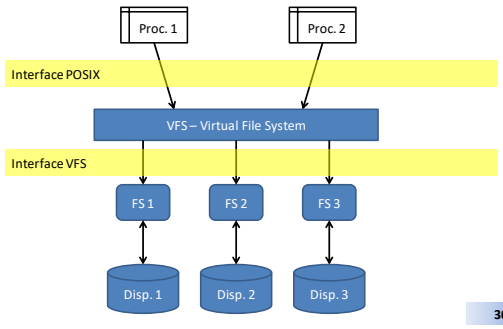
Sistemas de Arquivos Virtuais

- No UNIX é o contrário – Diferentes sistemas de arquivos são integrados em uma única estrutura;

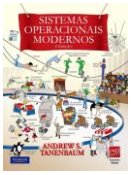
Ex: \ – **ext2**
\usr\ – **ext3**
\mnt\CD – **ISO 9660**

- Sun Microsystems – 1986 – VFS: Virtual File System;
- Busca integrar sistemas de arquivos distintos em uma estrutura ordenada;
- **IDEIA GERAL**: Abstrair a parte comum aos diferentes sistemas de arquivos e colocar o código em uma camada separada que chama o sistema de arquivos subjacentes para fazer o gerenciamento do dado;

Organização do VFS



Bibliografia - Básica



- 3ª Edição
- Páginas 169 – 180