Aula 15/10/2024

1. Introdução ao Gerenciamento de Arquivos

O gerenciamento de arquivos é uma das funções essenciais dos sistemas operacionais, responsável por organizar, armazenar e permitir o acesso eficiente aos dados em dispositivos de memória não volátil, como discos rígidos, SSDs e pendrives.

---

2. Componentes do Gerenciamento de Arquivos

- **Arquivo**:
  - Estrutura básica de armazenamento, com atributos como:
    - Nome
    - Tipo
    - Tamanho
    - Localização
  - Exemplo: `Trabalho_final.docx`.
- **Diretório**:
  - Estruturas hierárquicas que facilitam a organização e localização de arquivos.
- **Sistema de Arquivos**:
  - Implementa o conceito de arquivo, organizando o conteúdo em diferentes tipos:
    - NTFS (Windows)
    - Ext3/Ext4 (Linux)
    - FAT (Dispositivos portáteis)
- **Volumes**:
  - Divisões lógicas de um dispositivo físico, como as partições `C:` e `D:`.

---

3. Operações Básicas sobre Arquivos

- **Criar**: Alocar espaço no sistema de arquivos e criar uma entrada no diretório.
- **Abrir**: Verificar permissões e criar referência na memória principal.
- **Ler/Escrever**: Transferir dados entre o processo e o arquivo.
- **Fechar**: Liberar as estruturas de gerenciamento do arquivo.
- **Apagar**: Remover a entrada no diretório e liberar espaço.
- **Alterar Atributos**: Modificar nome, proprietário, ou permissões.

Exemplo de comandos no Linux:

bash $ echo "Um" > novo_arquivo.txt $ echo "Dois" >> novo_arquivo.txt

---

4. Formatos de Arquivos

- Sequência de Bytes: Forma mais simples; interpretação depende da aplicação.
- Texto: Usado para códigos, configurações. Exemplo: ASCII e UTF-8.
- Código Executável: Formatos como ELF (Linux) e PE (Windows), com seções específicas.

---

5. Formas de Acesso a Arquivos

- Acesso Sequencial: Operações em sequência do início ao fim. Exemplo: leitura linha a linha de um arquivo de log.
- Acesso Aleatório: Permite acessar posições específicas, sem percorrer todo o arquivo.
- Acesso Mapeado em Memória: Arquivo associado a um vetor na memória; muito usado por SOs para bibliotecas.

---

6. Compartilhamento de Arquivos

- Condições de disputa: Leituras concorrentes não apresentam problemas, mas escritas simultâneas podem causar inconsistências.
- Travas: - Exclusivas: garantem acesso exclusivo. -Compartilhadas: permitem múltiplas leituras simultâneas.

---

7. Controle de Acesso

```
Proprietário:
    Define quem tem direitos sobre o arquivo.
Permissões:
    Unix: leitura (r), escrita (w), execução (x).
    Exemplo no Linux:
```

---

8. Comandos Shell

```
$ ls: Lista arquivos no diretório.
$ chmod: Altera permissões.
$ stat: Exibe informações detalhadas de um arquivo.
$ grep: Busca por padrões em arquivos.

Exemplo:

$ grep -r "sistema" /etc/
```

---

9. Exercícios Resolvidos

```
Vantagem dos Diretórios: Facilita a organização e localização de arquivos.

Atributo Nome: Não é essencial para o sistema, mas ajuda o usuário.

Operações Básicas: Criar, abrir, ler, escrever, fechar, apagar, alterar atributos.

Descritores de Arquivos: Gerenciam as operações e armazenam atributos temporários.

Acesso Aleatório vs. Sequencial: Acesso aleatório é mais flexível; permite leitura/escrita
em qualquer posição.
```

File Systems in Operating Systems

Instructor: Marcelo Friske

Agenda:

- Disks and partitions/volumes
  - Block management
  - File allocation

Disk Space and Partitions

In operating systems, a disk is viewed as a large array of blocks (or sectors) with fixed sizes. These blocks are grouped into **partitions** or **volumes**, which are logical storage units within a physical device.

Storage Space is Divided Into:

1. **Configuration Area**:
   - Contains a partition table describing the partitioning of the device.
   - Holds the **Master Boot Record (MBR)**, which contains a small executable code to assist in booting the operating system.
2. **Partitions**:
   - Each partition contains a **Volume Boot Record (VBR)**, located at the beginning of the partition. This records the details of the partition and possibly holds boot code for the OS.
   - The rest of the partition consists of storage blocks for user files and directories.

Example of FAT Organization

- **Sector 0**: Boot Sector.
- **Sector 1 and 2**: FAT Location.
- **Sector 3 and 4**: Copy of FAT.
- **Sector 5 to 11**: Directory entries.
- **Sector 12 and onward**: File data.

Block Management

Block management is a layer in the OS responsible for interacting with device drivers and managing block data transfer. It maps **physical blocks** (512 or 4096 bytes in size) to **logical blocks** (a group of physical blocks).

- **Logical blocks** help simplify management and improve read/write performance by transferring larger groups of data. Logical block sizes can range from 4 KB to 64 KB.

Example:

- A 1 GB disk may contain over a billion 512-byte blocks.
- A logical block size of 32 KB can group 64 physical blocks for more efficient data transfer.

Logical Block Size and System Capacity

The logical block size directly affects the storage system's total capacity. For example, with FAT32:

- A 4 KB block size can address up to 1 TB.
- A 64 KB block size can address up to 16 TB.

Block Caching

Block caching is a mechanism where blocks are stored temporarily in **RAM** to improve performance. Since disk access is relatively slow, caching helps reduce the latency of read/write operations.

Caching Strategies

- **Read-through**: The requested block is delivered from cache if available. If not, it's read from disk and added to the cache for future requests.

- **Read-ahead**: In addition to the requested block, adjacent blocks are also loaded into cache to speed up sequential read operations.
- **Write-through**: Data is written directly to disk, with the cache maintaining a copy for subsequent access.
- **Write-back**: Data is initially written to cache and then committed to disk later. This boosts performance but poses a risk of data loss if a system crash occurs before the data is flushed to disk.

File Allocation Methods

File allocation is the process of assigning blocks to store file contents. The choice of allocation method impacts system performance and file organization.

1. Contiguous Allocation

Files are stored in sequential blocks. This provides fast access, but leads to **external fragmentation**, where there may be enough total space but not enough contiguous space to store large files.

Example:

- If a file requires 1 GB and is allocated contiguous 4 KB blocks, 262,144 blocks are needed.

Advantages:

- Fast sequential and random access.

Disadvantages:

- Low flexibility. Fragmentation makes it difficult to find large contiguous spaces, and files must be resized with care.

---

2. Chained Allocation

Each file block contains a pointer to the next block, creating a **linked list** of blocks. This avoids the need for contiguous space and eliminates external fragmentation.

Advantages:

- High flexibility; files do not need contiguous blocks.

Disadvantages:

- Slow random access, as each block must be read sequentially to find a specific part of the file.
- Vulnerable to errors: If one block becomes corrupted, the rest of the file becomes inaccessible.

FAT Allocation (File Allocation Table)

To improve chained allocation, FAT stores all block pointers in a **centralized table** rather than inside each block. This allows faster file access and better error handling.

---

3. Indexed Allocation

Rather than chaining blocks, indexed allocation uses an **index block** to store pointers to all the file's blocks. This allows for direct access to any block and eliminates both fragmentation and access delays caused by chaining.

Example:

- In UNIX, **i-nodes** are used to store file metadata and block pointers.

Advantages:

- Efficient access for both sequential and random operations.
- High robustness, as block errors do not affect other file blocks.

Disadvantages:

- Limited file size, as the number of block pointers in an index is fixed.

`Multi-level Indexed Allocation`

For larger files, multi-level indexing is used, where pointers can point to other blocks of pointers, forming a tree structure. This greatly increases the maximum file size.

Example:

- A system with 4 KB blocks and 32-bit pointers can store 1024 block pointers per block, allowing very large files to be managed.

Mounting and Unmounting Volumes

Before a volume can be used, it needs to be **mounted** by the OS. Mounting involves reading the volume's boot block, creating data structures in memory, and assigning an identifier to the volume for access.

- **Mounting on initialization**: Disk drives are mounted during system boot.
- **Mounting after initialization**: Removable media (e.g., USB drives) can be mounted manually after the system has started.

Unmounting releases all resources and closes all open files, safely removing the volume from the system.

Summary

- **File Systems** allow efficient management of files on disk, with partitions, volumes, and block management optimizing data access.
- **File Allocation** strategies (contiguous, chained, and indexed) play a critical role in file organization and system performance.
- **Caching** significantly improves performance by storing frequently accessed blocks in memory.
- **Mounting/Unmounting** volumes enables the system to dynamically manage different storage devices, ensuring data is accessible and safe during disconnection.

Aula 11 - Sistemas Operacionais: Sistemas de Arquivos

Professor: Marcelo Friske

Aula de hoje:

- Discos e partições/volumes
  - Principais elementos.
  - Gerência de blocos.
  - Alocação de arquivos.
- Exercícios.

Sistema de Arquivos: Partições

`Camadas de Implementação da Gerência de Arquivos`

Espaços de Armazenamento: Discos e Partições

- O sistema operacional vê um disco como um grande vetor de blocos (setores) de dados de tamanho fixo.
- O espaço de armazenamento é dividido em:
  1. Área de configuração reservada (início do disco):
     - Contém uma tabela de partições com informações sobre o particionamento.
     - Pequeno código executável (MBR - Master Boot Record) usado no boot.
  2. Uma ou mais partições:
     - **VBR** (Volume Boot Record) no início da partição.
     - Blocos de armazenamento disponíveis para armazenar arquivos.

Organização em Partições de um Disco Rígido: Layout

- **Super Bloco**: Parâmetros sobre o sistema de arquivos, carregado na memória.
- **I-nodes**: Estruturas de dados contendo informações sobre os arquivos.
- **Raiz**: Diretório raiz – árvore de diretórios.
- **Arquivos e Diretórios**: Armazenamento dos arquivos e diretórios.

Exemplo FAT (File Allocation Table)

- Setor 0: Setor de Boot.
- Setores 1 e 2: Localização da FAT.
- Setores 3 e 4: Cópia da FAT.
- Setor 5 a 11: Entradas de diretório.
- Setor 12 em diante: Dados.

Montagem de Volumes

- O SO acessa os arquivos de um volume:
  1. Leitura do bloco de inicialização.
  2. Criação de estruturas em memória para representar o volume.
  3. Definição de um identificador para o volume.
- Montagem durante inicialização (discos, drives) ou após (mídias removíveis).

`Desmontagem:`

- Fechamento de arquivos e remoção de estruturas de memória.

Gestão de Blocos

- Interação com drivers de dispositivos.
- Mapeamento de blocos físicos em blocos lógicos.
- Cache de blocos para otimizar operações de leitura e escrita.

Tamanho de Blocos Lógicos

- Blocos lógicos variam de 4KB a 64KB.
- Quanto maior o bloco, mais eficiente as operações de E/S, porém aumenta a fragmentação interna.

Alocação de Arquivos

- Função de alocar conteúdo e metadados de arquivos em blocos lógicos.

- Tipos de alocação:
  1. **Contígua**: Dados em blocos consecutivos.
  2. **Encadeada**: Cada bloco contém um ponteiro para o próximo.
  3. **Indexada**: Vetor de índices aponta para os blocos de arquivos.

Exemplo de Alocação Contígua

- Arquivo armazenado de forma sequencial em blocos consecutivos.
- Rapidez e robustez, porém baixa flexibilidade e fragmentação externa.

Alocação Encadeada: Simples e FAT

- Blocos formam uma lista encadeada.
- FAT (File Allocation Table) armazena os ponteiros dos blocos separadamente.

Alocação Indexada

- Uso de i-nodes para armazenar índices de blocos de arquivos.
- Multinível: Aumenta o tamanho máximo dos arquivos utilizando ponteiros indiretos.

Caching de Blocos

- **Read-through**: Lê bloco na cache, se não estiver, lê do disco.
- **Read-ahead**: Traz mais dados que a solicitação.
- **Write-through**: Escreve diretamente no disco.
- **Write-back**: Escreve na cache e posteriormente no disco.

Conclusão

- Sistemas de arquivos organizam e gerenciam o armazenamento de dados em discos e partições, com técnicas variadas de alocação e caching para otimizar o desempenho.