Sistemas de Arquivos em Linux e Windows

Introdução

Os sistemas de arquivos são a interface primária através da qual os usuários e as aplicações interagem com os dados armazenados em dispositivos de armazenamento persistente. Tanto o Linux quanto o Windows, como sistemas operacionais modernos e amplamente utilizados, implementam sistemas de arquivos robustos e ricos em funcionalidades, cada um com suas próprias arquiteturas, tipos de sistemas de arquivos suportados e APIs. Enquanto o capítulo anterior discutiu os conceitos gerais de sistemas de arquivos, este capítulo foca nas implementações específicas e nas características dos sistemas de arquivos encontrados nos ambientes Linux e Windows. Abordaremos os principais sistemas de arquivos nativos e suportados por cada SO, suas estruturas internas, as chamadas de sistema (APIs) para manipulação de arquivos e diretórios, e como eles gerenciam os dados no disco. Compreender essas especificidades é essencial para desenvolvedores, administradores de sistemas e qualquer pessoa que precise gerenciar dados ou otimizar o desempenho de E/S nessas plataformas.

Conceitos e Abstrações Comuns

Ambos os sistemas operacionais utilizam abstrações para permitir que diferentes tipos de sistemas de arquivos coexistam e sejam acessados de maneira uniforme pelas aplicações.

- **Linux: Virtual File System (VFS) ou Virtual Filesystem Switch:** O VFS é uma camada de abstração fundamental no kernel do Linux que fornece uma interface uniforme para todos os sistemas de arquivos. Ele define um conjunto comum de

operações (e.g., open, read, write, close) que as aplicações podem usar, independentemente do tipo de sistema de arquivos subjacente (e.g., ext4, XFS, NTFS, FAT). O VFS direciona essas chamadas para o driver do sistema de arquivos específico que gerencia o dispositivo de armazenamento.

- **Windows: Installable File System (IFS) Manager:** O Windows utiliza um mecanismo semelhante, onde diferentes drivers de sistema de arquivos (e.g., para NTFS, FAT32, ReFS, e sistemas de arquivos de rede) podem ser instalados e gerenciados pelo I/O Manager. Isso permite que o Windows suporte múltiplos sistemas de arquivos de forma transparente para as aplicações que usam a Win32 API para operações de arquivo.

Sistemas de Arquivos no Linux

O Linux é conhecido por seu suporte a uma vasta gama de sistemas de arquivos, desde os tradicionais sistemas de arquivos Unix até os mais modernos com journaling e funcionalidades avançadas.

Principais Sistemas de Arquivos Suportados/Nativos

- **ext (Extended File System) Family:**
  - **ext2:** Um dos primeiros sistemas de arquivos amplamente adotados no Linux. Robusto e estável, mas não possui journaling, o que pode levar a longos tempos de verificação (fsck) após uma falha no sistema.
  - **ext3:** Uma evolução do ext2 que adiciona journaling. O journaling melhora significativamente a velocidade de recuperação após falhas, registrando as alterações antes que sejam efetivamente escritas nos dados principais do sistema de arquivos. Oferece três modos de journaling: journal (dados e metadados), ordered (apenas metadados, mas garante que os dados sejam escritos antes dos metadados) e writeback (apenas metadados, com menor garantia de consistência em caso de falha).
  - **ext4:** O sucessor do ext3 e o sistema de arquivos padrão para muitas distribuições Linux. Inclui melhorias como suporte a volumes maiores e arquivos maiores, extents (para alocação contígua de blocos, melhorando o desempenho e reduzindo a fragmentação), alocação atrasada (delayed allocation), verificação de consistência mais rápida e journaling mais robusto.
- **XFS (Extent File System):** Um sistema de arquivos de alto desempenho com journaling, originalmente desenvolvido pela Silicon Graphics (SGI). É conhecido por sua escalabilidade, bom desempenho com arquivos grandes e operações paralelas de E/S. Utiliza alocação baseada em extents e árvores B+ para gerenciar espaço livre e inodes.
- **Btrfs (B-tree File System):** Um sistema de arquivos moderno com foco em funcionalidades avançadas, tolerância a falhas e facilidade de administração. Suporta copy-on-write (CoW) para todos os dados e metadados, snapshots (cópias instantâneas de baixo custo), subvolumes, RAID integrado, compressão transparente e verificação de integridade de dados.

- **Outros Sistemas de Arquivos:** O Linux também suporta muitos outros sistemas de arquivos, como JFS (Journaled File System da IBM), ReiserFS (conhecido por bom desempenho com arquivos pequenos, mas com desenvolvimento estagnado), F2FS (Flash-Friendly File System, otimizado para dispositivos de armazenamento baseados em flash NAND como SSDs), além de sistemas de arquivos de rede como NFS (Network File System) e CIFS/SMB (Common Internet File System/Server Message Block).

Estruturas de Dados Comuns no Linux (conceitos do VFS e implementações típicas)

- **Superblock (Superbloco):** Contém metadados críticos sobre o sistema de arquivos como um todo (e.g., tipo, tamanho, estado, número de blocos livres, número de inodes livres).
- **Inode (Index Node):** Uma estrutura de dados que armazena os atributos de um arquivo (e.g., permissões, proprietário, tamanho, timestamps) e os ponteiros para os blocos de dados do arquivo. Cada arquivo no sistema de arquivos é representado por um inode.
- **Dentry (Directory Entry):** Representa uma entrada de diretório, que mapeia um nome de arquivo para um inode. O kernel mantém um cache de dentries (dcache) para acelerar a tradução de nomes de caminho para inodes.
- **File Object (Objeto de Arquivo):** Representa um arquivo aberto por um processo. Contém informações como o ponteiro de posição atual no arquivo e o modo de acesso.

Chamadas de Sistema (APIs) Relevantes no Linux (Padrão POSIX)

O Linux implementa as chamadas de sistema padrão POSIX para operações de arquivo:

- open(): Abre ou cria um arquivo.
- read(): Lê dados de um arquivo.
- write(): Escreve dados em um arquivo.
- close(): Fecha um arquivo aberto.
- lseek(): Reposiciona o ponteiro de leitura/escrita do arquivo.
- stat(), fstat(), lstat(): Obtêm informações (atributos) sobre um arquivo.
- mkdir(): Cria um novo diretório.
- rmdir(): Remove um diretório (geralmente deve estar vazio).
- unlink(): Remove um nome (link) para um arquivo. Se for o último link, o arquivo é excluído.
- rename(): Renomeia ou move um arquivo/diretório.
- mount(): Monta um sistema de arquivos em um ponto específico da hierarquia de diretórios.
- umount(): Desmonta um sistema de arquivos.

Sistemas de Arquivos no Windows

O Windows tem seu próprio conjunto de sistemas de arquivos nativos, com o NTFS sendo o mais proeminente para instalações modernas.

Principais Sistemas de Arquivos Suportados/Nativos

- **NTFS (New Technology File System):** O sistema de arquivos padrão para todas as versões modernas do Windows (desde o Windows NT). É um sistema de arquivos robusto e rico em funcionalidades:
  - **Journaling:** Usa um log de transações ($LogFile) para garantir a rápida recuperação do sistema de arquivos em caso de falha.
  - **Segurança:** Suporta Listas de Controle de Acesso (ACLs) granulares para arquivos e diretórios, permitindo definir permissões detalhadas para usuários e grupos.
  - **Master File Table (MFT):** O coração do NTFS. A MFT é um arquivo que contém registros para todos os arquivos e diretórios no volume. Cada registro na MFT armazena os atributos do arquivo ou ponteiros para eles se forem muito grandes (atributos residentes vs. não residentes).
  - **Compressão e Encriptação:** Suporta compressão de arquivos e diretórios no nível do sistema de arquivos e Encrypting File System (EFS) para encriptação transparente.
  - **Alternate Data Streams (ADS):** Permite que múltiplos fluxos de dados sejam associados a um único nome de arquivo.
  - **Hard Links e Symbolic Links (Junction Points, Symbolic Links):** Suporta diferentes tipos de links.
  - **Suporte a Arquivos Grandes e Volumes Grandes.**
- **FAT (File Allocation Table) Family:**
  - **FAT12/FAT16:** Sistemas de arquivos mais antigos, usados no MS-DOS e versões iniciais do Windows. Limitados em tamanho de volume e arquivo.
  - **FAT32:** Uma extensão do FAT16 que suporta volumes maiores e arquivos de até 4GB. Comumente usado para dispositivos de armazenamento removíveis (pen drives, cartões de memória) devido à sua ampla compatibilidade entre diferentes sistemas operacionais.
  - **exFAT (Extended File Allocation Table):** Projetado pela Microsoft como um sucessor do FAT32, otimizado para dispositivos de armazenamento flash. Remove muitas das limitações do FAT32 (e.g., tamanho de arquivo e volume muito maiores) e é suportado por versões mais recentes do Windows, macOS e Linux.
- **ReFS (Resilient File System):** Introduzido com o Windows Server 2012, o ReFS foi projetado para maximizar a disponibilidade de dados, escalar eficientemente para grandes conjuntos de dados e fornecer integridade de dados (com verificação e correção de corrupção quando usado com Storage Spaces). Ele compartilha algum código com o NTFS, mas tem uma arquitetura diferente, focada na resiliência contra corrupção de dados.

Chamadas de Sistema (APIs) Relevantes no Windows (Win32 API)

O Windows fornece um conjunto abrangente de funções na Win32 API para operações de arquivo e diretório:

- CreateFile(): Abre ou cria arquivos, pipes, dispositivos, etc. Função muito versátil.
- ReadFile() e WriteFile(): Leem e escrevem dados de/para arquivos.
- CloseHandle(): Fecha um handle de arquivo aberto (ou qualquer outro objeto do kernel).
- SetFilePointer() ou SetFilePointerEx(): Reposiciona o ponteiro do arquivo.
- GetFileAttributesEx() e SetFileAttributes(): Obtêm e definem atributos de arquivo.
- CreateDirectory() e RemoveDirectory(): Criam e removem diretórios.
- DeleteFile(): Exclui um arquivo.
- MoveFile() ou MoveFileEx(): Renomeia ou move um arquivo/diretório.
- FindFirstFile(), FindNextFile(), FindClose(): Usadas para listar o conteúdo de um diretório.

Comparativo e Conclusão

Tanto Linux quanto Windows oferecem sistemas de arquivos sofisticados, cada um com suas forças. O Linux, através do VFS, proporciona grande flexibilidade com suporte a uma miríade de sistemas de arquivos, sendo ext4 e XFS escolhas populares para desempenho e confiabilidade, enquanto Btrfs oferece funcionalidades avançadas. O Windows foca primariamente no NTFS para suas instalações de sistema, um sistema de arquivos maduro e rico em recursos, complementado pelo FAT32/exFAT para interoperabilidade e pelo ReFS para cenários que exigem alta resiliência.

As APIs para manipulação de arquivos refletem as filosofias de design de cada sistema: o Linux adere ao padrão POSIX, promovendo portabilidade entre sistemas Unix-like, enquanto o Windows oferece a Win32 API, específica para sua plataforma.

Ambos os sistemas operacionais continuam a evoluir seus subsistemas de arquivos para lidar com as crescentes demandas de armazenamento, desempenho e novas tecnologias de hardware, como SSDs e armazenamento em nuvem. A escolha do sistema de arquivos apropriado e a compreensão de como interagir com ele através das APIs do sistema operacional são cruciais para o desenvolvimento de aplicações eficientes e para a administração eficaz dos dados.

Referências

- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley.
- Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson Education.

- Love, R. (2010). *Linux Kernel Development* (3rd ed.). Addison-Wesley Professional.
- Bovet, D. P., & Cesati, M. (2005). *Understanding the Linux Kernel* (3rd ed.). O'Reilly Media.
- Russinovich, M. E., Solomon, D. A., & Ionescu, A. (2017). *Windows Internals, Part 1* (7th ed.). Microsoft Press.
- Microsoft Docs. (Várias datas). *File Systems Documentation*. Recuperado de https://docs.microsoft.com
- The Linux Kernel Archives. (Várias datas). *Kernel Documentation*. Recuperado de https://www.kernel.org/doc/