Gerenciamento de Arquivos e Sistemas no Linux

Este documento visa consolidar as informações mais relevantes sobre o gerenciamento de arquivos, sistemas de arquivos e operações relacionadas em ambientes Linux, conforme as fontes fornecidas.

1. Conceitos Fundamentais de Sistemas de Arquivos

Sistemas de Arquivos (SA) são um dos conceitos mais importantes em sistemas operacionais, ao lado de processos (e threads) e espaços de endereçamento. Eles fornecem a abstração para o armazenamento persistente de dados em unidades de memória secundária, como discos rígidos e SSDs, tratando os dados como "sequências de bytes". O SO converte requisições de programas em transferências de blocos, abstraindo o tamanho do bloco para o programador.

Questões essenciais que surgem na gestão de grandes volumes de arquivos incluem: "Como você encontra informações?", "Como impedir que um usuário leia os dados de outro?" e "Como saber quais blocos estão livres?".

1.1 Tipos e Nomeação de Arquivos

- **Abstrações Principais**: Arquivos são unidades lógicas de informação criadas por processos, com um disco podendo conter milhões deles.
- **Nomeação**:
- Sistemas como UNIX (e OS X) distinguem entre letras maiúsculas e minúsculas (ex: "maria", "Maria", "MARIA" são arquivos distintos). O MS-DOS não faz essa distinção.
- Muitos sistemas operacionais aceitam nomes de duas partes separadas por um ponto, onde a parte após o ponto é a extensão (ex: prog.c).
- **Tipos de Arquivos (UNIX/Windows)**:
- **Arquivos Regulares**: Contêm informações do usuário, podendo ser ASCII (linhas de texto) ou binários.
- **Diretórios**: Arquivos de sistema usados para manter a estrutura hierárquica do sistema de arquivos.

- **Arquivos Especiais de Caracteres**: Relacionados a entrada/saída serial (terminais, impressoras, redes).
- **Arquivos Especiais de Blocos**: Usados para modelar discos.
- **Acesso a Arquivos**: Com o uso de discos, tornou-se possível o "acesso aleatório", onde bytes ou registros podem ser lidos fora de ordem, ou acessados por chave.

2. Implementação e Organização do Sistema de Arquivos

Sistemas de arquivos são armazenados em discos, que podem ser divididos em partições independentes. O Setor 0 do disco é o **MBR (Master Boot Record)**, usado para inicializar o computador.

2.1 Estruturas Essenciais na Partição

Ao criar ou formatar uma partição com um sistema de arquivos, são estabelecidas estruturas para identificar blocos livres e alocados, e para representar arquivos e os blocos que eles utilizam.

- **Superbloco**: Contém parâmetros-chave do sistema de arquivos e é lido para a memória na inicialização ou no primeiro acesso.
- **Mapa de Bits ou Lista de Ponteiros**: Informações sobre blocos disponíveis no sistema de arquivos.
- **Diretórios e Arquivos**: O restante do disco contém os dados propriamente ditos.

2.2 Métodos de Alocação de Blocos

A forma como os blocos são alocados para os arquivos é crucial para o desempenho e a gestão do espaço.

- **Alocação Contígua**: Blocos consecutivos são alocados. "O desempenho da leitura é excelente, pois o arquivo inteiro pode ser lido do disco em uma única operação." No entanto, sofre de fragmentação com o tempo.
- **Lista Encadeada**: Blocos podem estar dispersos, com cada bloco contendo um ponteiro para o próximo. "Nenhum espaço é perdido para a fragmentação de disco." Desvantagens incluem acesso aleatório lento e o espaço do ponteiro que não pode ser usado para dados.

- **Lista Encadeada com Tabela na Memória (FAT)**: Melhora o acesso aleatório mantendo a tabela de alocação de arquivos (FAT) na memória. A principal desvantagem é que "a tabela inteira precisa estar na memória o todo o tempo para fazê-la funcionar."
- **I-nodes (index-node)**: Associa cada arquivo a uma estrutura de dados que lista os atributos e endereços de disco dos blocos. A vantagem é que "o i-node precisa estar na memória apenas quando o arquivo correspondente estiver aberto." Para arquivos grandes, os i-nodes podem apontar para blocos que contêm mais endereços de blocos de dados (indireção).

2.3 Gerenciamento de Espaço Livre

Dois métodos principais para monitorar blocos livres são:

- **Lista Encadeada de Blocos Livres**: Blocos de disco contêm números de blocos livres.
- **Mapa de Bits**: Um bit para cada bloco do disco, indicando se está livre ou ocupado. Um disco com *n* blocos requer *n* bits.

2.4 Implementação de Diretórios

Diretórios são arquivos que armazenam atributos de outros arquivos. Uma forma é armazená-los diretamente na entrada do diretório, contendo nome, atributos e endereços de disco dos blocos.

3. Gerenciamento de Sistemas de Arquivos no Linux

Linux suporta diversos tipos de sistemas de arquivos, locais e remotos, através de uma camada de abstração chamada **Virtual File System (VFS)**. Isso permite que aplicações usem a mesma sintaxe e chamadas de sistema para leitura e escrita, independentemente do tipo de SA subjacente.

3.1 Comandos Essenciais de Gerenciamento

A manipulação de um sistema de arquivos envolve as seguintes etapas e comandos:

- **Criar Partições**: fdisk (gerencia partições).

- **Formatar Partições / Criar SA**: mkfs (cria um sistema de arquivos, ex: mke2fs).
- **Montar SA**: mount (torna o SA acessível em um subdiretório da árvore lógica do SO).
- **Verificar Consistência**: fsck (verifica e repara inconsistências).
- **Áreas de Swap**: swapon, swapoff (ativa/desativa partições como espaço auxiliar de memória virtual).
- **Volumes Lógicos (LVM)**: pvcreate, vgcreate, lvcreate (gerenciamento flexível de espaço em disco).
- **Consulta de Status**:
- stat: exibe status de arquivo ou sistema de arquivos (stat -f para informações do SA).
- df: reporta o uso de espaço em disco (df -lh para leitura humana, df -lhi para inodes).
- dumpe2fs ou tune2fs: ajusta parâmetros de sistemas de arquivos ext2/ext3/ext4.

3.2 Estrutura Hierárquica Unix (FHS)

O sistema de arquivos Unix é baseado em uma única estrutura hierárquica (árvore) de diretórios, começando pela raiz /.

- /: Diretório raiz.
- **/etc**: Arquivos de configuração.
- **/bin**: Utilitários de uso geral.
- **/sbin**: Utilitários de administração (alguns restritos ao root).
- **/dev**: Arquivos especiais que representam dispositivos, criados com mknod. Contêm major number (driver) e minor number (dispositivo específico).
- **/usr**: Hierarquia secundária (incluindo /usr/lib e /usr/include).
- **/lib**: Bibliotecas compartilhadas essenciais.
- **/home**: Área de trabalho dos usuários.
- **/var**: Área para spool de impressão, e-mails e logs.
- **/boot**: Arquivos para inicialização do sistema.
- **/mnt**: Diretório para sistemas de arquivos temporários montados.

4. Operações de Leitura/Escrita e Desempenho

O SO interage com os controladores de disco para converter requisições de programas em transferências de blocos.

4.1 Buffering e Caching

- **Nível do SO**: Em operações de leitura (read(2)), o sistema de arquivos lê um bloco inteiro do disco (mesmo se poucos bytes são solicitados) e o armazena em uma área de **buffer ou cache na memória**. Requisições subsequentes para dados no cache são mais rápidas.
- **Otimização de Gravações**: O SO pode reordenar gravações pendentes para otimizar o acesso ao disco, agrupando aquelas que se referem a áreas próximas, diminuindo atrasos de movimentação dos braços (em HDDs).
- **Consistência**: O adiamento das gravações pode levar a inconsistências em caso de falta de energia, se os dados em cache não forem persistidos. Por isso, é crucial "informar o SO antes de remover os dispositivos de armazenamento USB" para que dados pendentes sejam gravados.
- **Escrita Síncrona**: Para garantir consistência imediata, operações de escrita podem ser configuradas como síncronas (O_SYNC ou O_DIRECT flags em open), onde a operação só retorna após a gravação completa.

4.2 Interfaces de Programação (C/POSIX)

- **Chamadas de Sistema (System Calls)**: open(2), read(2), write(2), close(2), lseek(2) (reposiciona offset), unlink(2) (remove arquivo), fsync(2) / fdatasync(2) (sincroniza dados com disco), truncate(2) (redimensiona arquivo).
- open retorna um int (descritor de arquivo).
- openat(): Permite evitar condições de corrida ao abrir arquivos em diretórios diferentes do atual, usando um descritor de diretório.
- **Interface ANSI C (Funções de Stream)**: fopen(3), fread(3), fwrite(3), fclose(3).
- fopen retorna um FILE * (ponteiro para uma estrutura FILE).
- fopen provê **E/S bufferizada**, o que pode melhorar o desempenho em acessos sequenciais, e é padronizada, tornando programas portáveis.
- fread() internamente usa read() do SO, mas faz solicitações em múltiplos do tamanho dos blocos de disco (ex: 4KB).

- **Mapeamento de Memória (mmap)**: Permite mapear offsets de um arquivo para uma área da memória virtual do processo. O acesso aos dados mapeados é feito via ponteiros de memória, com o SO gerenciando a paginação e busca no disco.

4.3 Operações de E/S Avançadas

- **Leitura Não Bloqueante (O_NONBLOCK)**: Permite que open() e operações subsequentes não causem o processo a esperar. Se dados não estão disponíveis, read() retorna -1 e errno é EAGAIN.
- **Monitoramento de E/S (select)**: select(2) permite monitorar múltiplos descritores de arquivo (ou sockets) para disponibilidade de leitura, escrita ou mudanças de estado, com um timeout opcional.
- **E/S Vetorizada (readv, writev)**: Permitem ler ou escrever dados de/para múltiplos buffers com uma única chamada de sistema (scatter/gather I/O), otimizando transferências.

5. Segurança e Gerenciamento de Acesso

A segurança no Unix é baseada em permissões atribuídas a um proprietário, um grupo e outros usuários, controladas por chown, chgrp e chmod.

5.1 Permissões Tradicionais (rwx)

- **ls -l**: Exibe 10 colunas de controle, indicando tipo de entrada (diretório d, link l, dispositivo de bloco b, etc.) e 9 colunas de permissões (3 grupos de 3: proprietário, grupo, outros).
- **Permissões de Arquivos**: r (leitura), w (escrita), x (execução).
- **Permissões de Diretórios**: r (listar conteúdo), w (criar/remover/renomear), x (entrar no diretório).
- **chmod**: Ajusta direitos de acesso (apenas root e proprietário). Notação simbólica (u+s, o-w) ou octal (ex: chmod 754 arq -> rwxr-xr--).
- **chown**: Altera proprietário (somente root).
- **chgrp**: Altera grupo (usuários podem alterar grupos de arquivos de sua propriedade para grupos aos quais pertencem).

- **umask**: Define as permissões a serem *removidas* por padrão em novos arquivos/diretórios criados. Ex: umask 022 remove permissão de escrita para grupo e outros.

5.2 Atributos Especiais (setuid, setgid, sticky bit)

- **setuid (u+s ou 4xxx)**: Quando aplicado a um executável, o processo resultante herda o UID do proprietário do arquivo, não do usuário que o invocou (ex: ping).
- **setgid (g+s ou 2xxx)**:
- Em executáveis: Processo herda o GID do grupo do arquivo.
- Em diretórios: Novos arquivos criados nesse diretório herdam o grupo associado ao diretório, não o grupo padrão do usuário.
- **sticky bit (+t ou 1xxx)**: Em diretórios (ex: /tmp), permite que qualquer usuário crie arquivos, mas somente o proprietário do arquivo ou o root pode remover/renomear os arquivos dentro desse diretório, evitando exclusões acidentais por outros usuários.

5.3 Gerenciamento de Usuários e Senhas

- **Usuários Locais**: Definidos em /etc/passwd.
- **Grupos**: Definidos em /etc/group.
- **Senhas**: Armazenadas em /etc/shadow (com permissões restritivas ----------).
- O arquivo /etc/shadow contém uma senha criptografada que inclui um "sal" (valor aleatório único para cada senha) para impedir que senhas idênticas resultem na mesma criptografia.
- A senha real não é armazenada. Na autenticação, o SO gera a criptografia da senha fornecida com o "sal" armazenado e compara o resultado.
- **Comandos de Gerenciamento**: useradd, groupadd, usermod, userdel, groupdel, groupmod.

5.4 Mudança de Privilégios

- **root**: Usuário com UID 0 e privilégios totais.
- **su**: Altera a identificação do usuário na sessão de shell atual (requer senha do usuário alvo, ou root se não especificado).

- **sudo**: Permite a execução de comandos específicos com privilégios de superusuário, sem mudar a sessão completa. A configuração é feita em /etc/sudoers e requer a senha do usuário que está executando o sudo.