|  |  |
| --- | --- |
|  | **FIPP – SISTEMAS DE INFORMAÇÃO – PRESIDENTE PRUDENTE.** |

**ANDRESSA HISAE TSUKASAKI – R.A. 261742078  
GABRIEL MIGUEL NAVAS – R.A. 261741888  
GUSTAVO DILLIO LEITÃO – R.A. 261741748**

**QUESTÕES REFERENTE A GERENCIAMENTO DE ARQUIVO – PARTE 2**

Presidente Prudente - SP

2019

**Questões referente a Gerenciamento de Arquivo - Parte 2**

1. **Explique as diferenças entre as duas soluções de implementação de diretórios adotadas para entradas (nomes) de tamanho variados.**

• **Primeira Solução:** Em linhas, ou seja, o cabeçalho tem o tamanho da entrada, nome do proprietário, horário de criação, proteção e nome do arquivo de tamanho indeterminado como parte fixa, finalizado por um caractere especial. Sua desvantagem é lacuna de tamanho variado no diretório, podendo não caber um novo arquivo, mas a compactação do diretório pode ser a solução, pois ele está na memória;

• **Segunda Solução:** Entradas de tamanho fixo e juntar nomes dos arquivos em uma área temporária (heap) no final do diretório. Sua vantagem é que não há lacunas e sempre caberá um novo arquivo, não é necessário completar os nomes com caracteres especiais e utiliza algoritmos de tabela hash para agilizar o processamento de busca.

1. **Quais as principais estruturas de entradas em diretórios empregadas em sistemas operacionais CPM, UNIX e Windows?**

**• CP/M:** Existe somente um diretório que contém entradas de tamanho fixo. Em ordem, o primeiro byte é o código do usuário, 8 bytes para nome da entrada, 3 bytes para o tipo de extensão do arquivo, 1 byte que aponta para extensão e no final 16 blocos.

**• WINDOWS:** Utiliza um sistema para nome longos, se um arquivo tem nome longo, será armazenado em uma ou mais entradas de diretórios. A estrutura em sequência é descrita: 8 bytes para nome do arquivo, 3 bytes para extensão, 1 para atributos, 10 bytes para data de criação e último acesso, 4 bytes para última escrita e 4 bytes para tamanho do arquivo.

**• UNIX:** Sistema de arquivo na forma de árvore iniciando-se no diretório raiz. Com adições de ligações, formando um grafo orientado acíclico. Possui um campo de 2 bytes para o número do i-node do arquivo e 14 bytes para o nome do arquivo. No i-node é armazenado os atributos como tamanho do arquivo, data da criação, último acesso, última alteração, proprietário, grupo, proteção e contador do número de entradas no diretório que apontam para o i-node, os 12 primeiros blocos diretos.

1. **Explique as diferenças entre link simbólico e link físico.**

**• Link simbólico:** Arquivos que funcionam como atalhos ou referência a outro arquivo ou diretório, ou seja, contém apenas o caminho do arquivo ao qual ele está ligado. Desta forma, somente o verdadeiro proprietário tem um ponteiro para i-node e o usuário tem apenas o caminho para o arquivo. Se o arquivo for destruído, dá falha no acesso. Sua desvantagem é o número excessivo de acesso a disco para resolver o caminho, porém é uma boa maneira de economizar espaço em disco e recursos do sistema. Exemplo: diretório e:/isos para o destino c:/isos  mklink /D "e:\isos" "c:\isos".

**• Link físico:** Compartilham o mesmo i-node do arquivo. Só podem referenciar arquivos, e nunca diretórios, não podem referenciar arquivos em outros volumes ou partições, hardlinks continuam funcionando mesmo que o arquivo original tenha sido apagado, e fazem referência ao i-node do arquivo original, ao contrário dos softlinks, que referenciam nomes de arquivos e diretórios, os hardlinks referenciam uma posição física da partição. Sua desvantagem é que a criação de uma ligação não altera a propriedade, só aumenta o contador de ligações, ou seja, se o dono apagar o arquivo e zerar o i-node, o usuário fica com ponteiro errado. Exemplo: ln caminho/arquivo-original caminho/link.

1. **Explique como é efetuada a gerência de espaço livre através de mapas de bits e lista ligada.**

• **Lista ligada:** lista encadeada de blocos, com cada bloco contento tantos blocos livres quantos couberem nele. Exemplo: com um bloco = 1 KB e com blocos de disco de 32 bits, cada bloco na lista de blocos livres contém os números de 256 blocos livres. Ela mantém somente um bloco de ponteiros para blocos livres na memória e quando um arquivo é criado, os blocos necessários são retirados do bloco de ponteiros. Quando esse bloco se esgota, um novo bloco de ponteiros é lido do disco e quando um arquivo é removido, seus blocos liberados são incluídos no bloco de ponteiros. Quando cheios são escritos no disco. Sua desvantagem é que o número de E/S é alto quando o bloco está quase cheio, pois quando enche o bloco, escreve no disco, se precisar de blocos, lê do disco.

**• Mapa de bits**: blocos representados por 1 (livre) ou 0 (alocado), dependendo de n blocos do disco requer um mapa de n bits. Ocupa menos espaço que as lista ligadas, pois usa apenas 1 bit por bloco, enquanto a lista usa 32 bits por bloco. Com mapas de bits também é possível manter um bloco de memória e usar o disco somente quando o bloco estiver cheio ou vazio. Sua vantagem são as alocações em um bloco único de mapa de bits que faz com que os blocos de disco fiquem próximos uns dos outros, minimizando os movimentos dos braços de disco.

1. **Quando se fala em Desempenho do Sistema de Arquivos, o uso de CACHE DE BLOCOS ou DE BUFFER é uma boa solução. Explique o funcionamento de um cachê (sua operação).**

É uma técnica empregada que mantêm blocos mais utilizados na memória, é de fato mais rápido buscar um bloco na memória do que em disco, com isso a leitura é satisfatoriamente rápida. Vários algoritmos podem ser usados para gerenciar uma cache, um em comum é conferir todas solicitações para ver se o bloco necessário está no cache. Se estiver, o pedido de leitura pode ser satisfeito sem acesso ao disco. Se o bloco não estiver, primeiro ele é lido no cache e então copiado para onde quer que seja necessário. Solicitações subsequentes para o mesmo bloco podem ser satisfeitas a partir do cache. Se o cache estiver cheio, algum bloco deve ser removido para que um novo bloco possa entrar.

1. **Apresente a arquitetura geral de gerência de arquivos presente em um sistema operacional qualquer, explicando seus principais elementos constituintes (um sistema de arquivo é formato por três componentes principais).**

* **Implementação de diretório:** Existem tipos de diretórios – único (um diretório com seus arquivos), de dois níveis (um diretório com outros diretórios com seus arquivos) e hierárquico (árvore de diretórios) – quanto ao hierárquico, há maneiras de especificar nomes dos arquivos – **nome de caminho absoluto** (consiste no caminho do diretório-raiz para o arquivo) e **nome de caminho relativo** (designação de um diretório como o de trabalho atual). Esses nomes localizam a **entrada de diretório**, que possuem, dependendo do sistema, o endereço do disco do arquivo inteiro (alocação contígua), o número do primeiro bloco (lista encadeada) ou o número do i-node (Linux/Unix). O SO que aceita um sistema de diretório hierárquico tem duas entradas especiais em cada diretório, “.” e “..”. O primeiro refere-se ao diretório atual e o segundo ao pai (exceto no diretório-raiz, onde ele refere-se a si mesmo). Algumas das operações que podem ser realizadas nos diretórios são create, delete, opendir, closedir, readdir, rename, link e unlink, na qual link pode ser o link físico (cria uma ligação do arquivo existente para o nome especificado pelo caminho) ou simbólico (nome criado que aponta para um outro arquivo).
* **Implementação do arquivo:** Sistemas de arquivos são armazenados em discos e sua maioria pode ser dividida em uma ou mais partições, com sistemas de arquivos independentes em cada partição. Na implementação, a questão mais importante é controlar quais blocos de disco vão com quais arquivos, e para isso há métodos diferentes – Alocação contígua, lista encadeada, lista encadeada na RAM e i-nodes.
* **Gerenciamento de espaço em disco:** Como os arquivos costumam ser armazenados em disco, torna-se necessário o gerenciamento de espaço de disco, tendo duas estratégias para armazenar um arquivo de n bytes – alocar n bytes consecutivos de espaço ou dividir o arquivo em uma série de blocos (não necessariamente) contíguos. O tamanho do bloco implica diretamente no desempenho – unidade de alocação grande pode desperdiçar espaços internos do bloco (arquivo pequeno não ocupar todo o bloco), porém é bom para o desempenho pela pouca quantidade de blocos para leitura, já a unidade de alocação pequena é o inverso, é necessário fazer a leitura de muitos blocos e isso é custoso, mas a ocupação no disco é uma vantagem. Para monitorar os blocos livres, tem-se dois métodos – lista ligada (cada bloco da lista contém a quantidade de blocos livres que cabem nele) e mapa de bits (contém a mesma quantidade de blocos do disco e representa 1 como livre e 0 como ocupado, um bit por bloco). Para evitar que haja exagero no uso do espaço de disco, existem algumas quotas do disco por usuário – ao abrir um arquivo, os atributos e endereços de disco são localizados e colocados em uma tabela de arquivos aberta na memória principal, entre os atributos há uma entrada dizendo quem é o proprietário e quaisquer aumentos no tamanho do arquivo serão cobrados da cota do proprietário. Uma segunda tabela contém os registros de cotas de todos os usuários com um arquivo aberto, mesmo que esse arquivo tenha sido aberto por outra pessoa em uma tabela de arquivos aberta na memória principal, entre os atributos há uma entrada dizendo quem é o proprietário e quaisquer aumentos no tamanho do arquivo serão cobrados da cota do proprietário. Uma segunda tabela contém os registros de cotas de todos os usuários com um arquivo aberto, mesmo que esse arquivo tenha sido aberto por outra pessoa.

1. **Do ponto de vista lógico, quais as principais diferenças entre a estrutura de diretórios Unix e do Windows?**

Toda estrutura de atributos do Windows fica na própria estrutura de diretórios, como também o tipo do arquivo. Já no Linux, fica apenas o nome e o endereço de onde se encontra os atributos e blocos de dados.

1. **Explique os tipos de referências possíveis a arquivos em uma estrutura de diretórios. Lembre-se dos registros utilizados nas estruturas de diretórios.**

Na primeira solução, os blocos de disco não são listados em diretórios, mas em uma pequena estrutura de dados associada com o arquivo em si. Os diretórios apontariam então apenas para a pequena estrutura de dados. Essa é a abordagem usada em UNIX (em que a pequena estrutura de dados é o i-node).

Na segunda solução, B se liga a um dos arquivos de C, obrigando o sistema a criar um novo arquivo do tipo LINK e a inseri-lo no diretório de B. O novo arquivo contém apenas o nome do caminho do arquivo para o qual ele está ligado. Quando B lê do arquivo ligado, o sistema operacional vê que o arquivo sendo lido é do tipo LINK, verifica seu nome e o lê. Essa abordagem é chamada de ligação simbólica, para contrastar com a ligação (estrita) tradicional.

1. **Enumere alguns** **problemas que ainda precisam ser resolvidos na implementação de um sistema de arquivos.**
2. **Insegurança na consistência**: Muitos sistemas de arquivos leem blocos, modificam-nos e só depois os escrevem. Se o sistema cair antes de todos os blocos modificados terem sido escritos, o sistema de arquivos pode ser deixado em um estado inconsistente. O problema é especialmente crítico se alguns dos blocos que não foram escritos forem blocos de i-nodes, de diretórios ou blocos contendo a lista de blocos livres.
3. **Backup**: É possível se encontrar em situações que pode perder ou alguém pegar o dispositivo que foi usado para realizar o backup, ou ao realizar um backup diário e acontecer do fogo que queimar os computadores e também queimar todos os discos de backup.
4. **Desperdício do disco**: Sistema de arquivo que aloca blocos com um tamanho muito grande tende a ter a fragmentação interna em disco, o que é muito ruim, pois com isso perde espaço em disco. Como também a alocação contígua, com o tempo a remoção de blocos, vai criando ficando lacunas em disco. O armazenamento de arquivo com lista ligada em memória, Windows utiliza isso, perde-se case 1,5gb em memória principal somente com metadados e diretórios.

1. **Armazenamento contíguo**: Armazenar um arquivo como uma sequência contígua de bytes tem o problema óbvio de que se um arquivo crescer, ele talvez tenha de ser movido dentro do disco.
2. **Os discos CD-ROM e CD-R não permitem que seus arquivos sejam modificados. Qual estratégia de alocação de arquivos é recomendável para o sistema de arquivos deste tipo de mídia de armazenamento? Justifique sua resposta.**

É recomendado o tipo ISO 9660, pois é o tipo padrão utilizado mundialmente, implementação onde os blocos são preenchidos sequencialmente em espiral. A um tempo atrás a Microsoft criou um padrão chamado Juliet contendo atributos a mais em sua estrutura, mas não deu certo, pois o mercado já utilizava o padrão ISO 9660.