Sistemas Operacionais

Sistema de Arquivos

Introdução

- Todas as aplicações precisam armazenar e recuperar informações.
- Considerações sobre os processos:
  - Espaço próprio de endereçamento limitado;
  - Quando um processo termina, suas informações são perdidas;
  - Problemas com compartilhamento, pois o espaço de endereçamento é restrito.
- Requisitos essenciais para o armazenamento de informações:
  - ser possível armazenar uma quantidade grande de informações;
  - a informação deve sobreviver ao término do processo;
  - vários processos podem acessar a informação concorrentemente.

2

Introdução

- Solução
  - manter a informação armazenada em disco ou em qualquer outro dispositivo externo de armazenamento, em unidades denominadas arquivos.
- Arquivo
  - mecanismo de abstração que fornece uma forma de armazenar e recuperar dados em disco.
  - *Persistente* → um arquivo só pode desaparecer quando seu proprietário assim desejar.
- Sistema de Arquivos
  - Parte mais visível do SO, responsável pela manipulação de arquivos e diretórios.
  - O SO cria um recurso lógico a partir de recursos físicos existentes no sistema computacional.

* **Partição**: abstração que permite – a partir do disco físico – criar discos lógicos.

3

Arquivos

- Identificação
  - Nome do arquivo (seqüência de caracteres cujo formato varia conforme o SO)
  - Exemplos....
- Tipos
  - Arquivos regulares (ordinários)
    - Contém informações do usuário (ASCII ou BIN)
  - Diretórios
    - Conjuntos de referências a arquivos
  - Especiais de caracteres
    - Modelagem de dispositivos de E/S seriais (impressoras, redes)
  - Especiais de blocos
    - Modelagem de discos

* Em UNIX, arquivos especiais estão no diretório ***/dev***

4

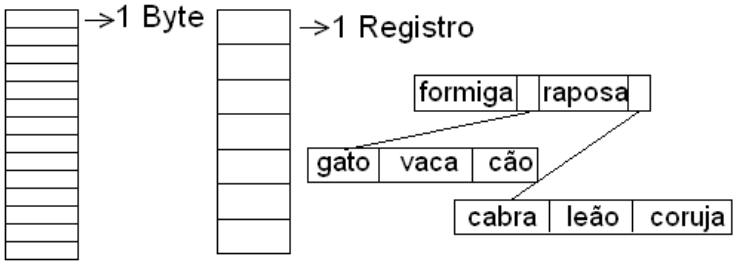
Arquivos

- Atributos
  - Tipo do conteúdo, tamanho, data e hora (acesso e alteração), ID do usuário, lista de usuários que podem acessá-lo, etc.
- Operações básicas
  - Criação/destruição do arquivo, leitura/alteração do conteúdo, execução do programa, alteração de atributos, etc.
- Métodos de Acesso
  - Seqüencial: conteúdo é lido byte a byte, seqüencialmente.
    - Garante flexibilidade (compiladores, impressoras, backups, etc.)
  - Acesso relativo (seqüência de registros de tamanho fixo)
    - Inclui na chamada de sistema qual a posição do arquivo a ser lida.
    - Registro 1 ocupa os bytes 0 a (TAM-1)
    - Outra abordagem: uso de syscall adicional para o posicionamento.

5

Arquivos

- Acesso relativo (continuação)
  - Usado em sistemas antigos (entrada feita por cartões e saída em impressoras de colunas)
- Árvore
  - Usado em sistemas comerciais;
  - Busca de registros baseada em chaves.



6

Implementação de arquivos

- Forma básica:
  - Para cada arquivo é criado um **descritor de arquivos**.
    - Registro onde são mantidas informações sobre o arquivo (atributos e outros dados não visíveis)
    - Enquanto o arquivo é acessado, seu descritor de arquivo é constantemente necessário.
    - Ex.: na leitura/gravação, o DA é lido para determinar a localização no disco dos dados.
    - Leitura do DA a partir do disco é inviável: questões de desempenho.
    - Solução: manter em memória uma tabela contendo todos os descritores de arquivos em uso (manter consistência: o SO exige que os programas informem quando o arquivo está em uso).

7

Implementação de Arquivos

- O SA utiliza uma Tabela dos D.A. Abertos (TDAA)
  - Mantém informações relativas aos arquivos abertos por todos os processos.
  - Possibilita que vários processos abram um mesmo arquivo simultaneamente.
  - Etapas (exemplificando a chamada de sistema open)
    - Localiza em disco o DA cujo nome foi fornecido (pesquisa baseada em partição/diretórios -*lookup*). Sucesso → <nº da partição, end. do DA>
    - Verifica se o arquivo solicitado já está aberto (pesquisa na TDAA).
    - Caso o arquivo não se encontre aberto, é alocado uma entrada livre e campos adicionais em memória são inicializados.
    - Através de uma consulta no próprio descritor, é verificado se o usuário tem direito de acesso ao arquivo.
    - Quando o processo realiza uma chamada *close*, decrementa-se o número de processos que estão usando o arquivo. Chegando a zero, o DA é atualizado em disco e a entrada na TDAA é liberada.

8

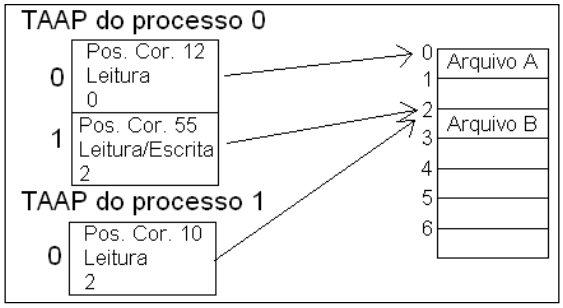
Implementação de Arquivos

- As informações da TDAA não variam conforme o processo que está acessando o arquivo.
- Porém, certas informações estão relacionadas unicamente a determinado processo.
  - Ex.: Posição corrente no arquivo.
  - Solução: Criar para cada processo uma Tabela de Arquivos Abertos por Processo (TAAP)
    - Cada processo possui sua TAAP;
    - Cada entrada da TAAP corresponde a um arquivo aberto pelo processo correspondente.
    - Para cada entrada → posição corrente no arquivo, tipo de acesso e apontador para a entrada correspondente na TDAA.

9

TDAA e TAAP

- Tanto a TDAA como as TAAP devem ficar na memória do SO, fora do acesso dos processos de usuário.



- Uma vez aberto o arquivo, a cada read/write não é conveniente fornecer o nome do arquivo e sim um número da entrada na TAAP associada com o arquivo.
- Muitos SO chamam esse número de *handle* do arquivo.

10

Diretórios

- Maneira que o SA organiza e controla os arquivos.
- Sistema de diretório em nível único
  - Um diretório mantém todos os arquivos;
  - Simples de implementar e rápido para localizar arquivos;
  - Esquema não mais empregado em sistemas multiusuários.
- Sistema de diretório em dois níveis
  - Diretório privado para sistema e para cada usuário;
  - Implicitamente, o sistema sabe de qual usuário é o arquivo referenciado.

  open ("x") ... open ("/joão/x")
- Sistema de diretório hierárquicos
  - Árvore de diretórios, cujos diretórios podem conter subdiretórios;
  - Sistema de arquivos modernos são assim organizados.

11

Diretórios

- Nomes de caminhos
  - Necessidade de especificar o nome dos arquivos
  - Nome do caminho absoluto
    - Caminho único entre o diretório raiz e o arquivo;
    - Separador (/, \, >)
  - Nome do caminho relativo
    - Considera o diretório atual para referenciar o arquivo;
    - Geralmente mais conveniente.

    Ex.: (diretório corrente /home/alunos)

    cp /home/alunos/programa.c /home/alunos/programa.bak

    cp programa.c programa.bak
    - Entradas especiais (. e ..)

    cp ../admin/fonte.c .
- Operações com diretórios
  - Maior dependência do sistema: create, delete, opendir, closedir, readdir, ,rename, link e unlink.

12

Implementação de Diretórios

- Ao abrir um arquivo, o SO usa o caminho para localizar a entrada no diretório.
- A entrada no diretório fornece informações para localizar os blocos de disco.
  - Endereço de disco do arquivo (alocação contígua); O número do 1º bloco (listas ligadas); O número do i-node (nós índice).
  - Necessidade de armazenar os atributos
  - diretamente na entrada do diretório;
  - usar um nó para tais informações.

→ Estratégias convenientes para nome de arquivos curtos e tamanho fixo.

  - Sistemas atuais suportam nomes longos com tamanhos variáveis.
  - Reservar 255 caracteres por entrada de diretório

    → simples, porém ineficiente.

13

Implementação de Diretórios

- Entradas de diretório com tamanho variável
  - Linear: Parte fixa (atributos comuns com um formato fixo) mais um nome real para o arquivo (por mais longo que ele seja)

    → lacunas variáveis e tamanho potencialmente grande.
  - *Heap:* Entradas fixas com nomes agrupados em uma área temporária (*heap*)

    → qualquer entrada removida está apta a receber outro arquivo;

    → gerência do *heap*.
- **Independente do projeto**: busca linear de arquivo no diretório.

  → Tabelas *hash* ou cache de buscas.

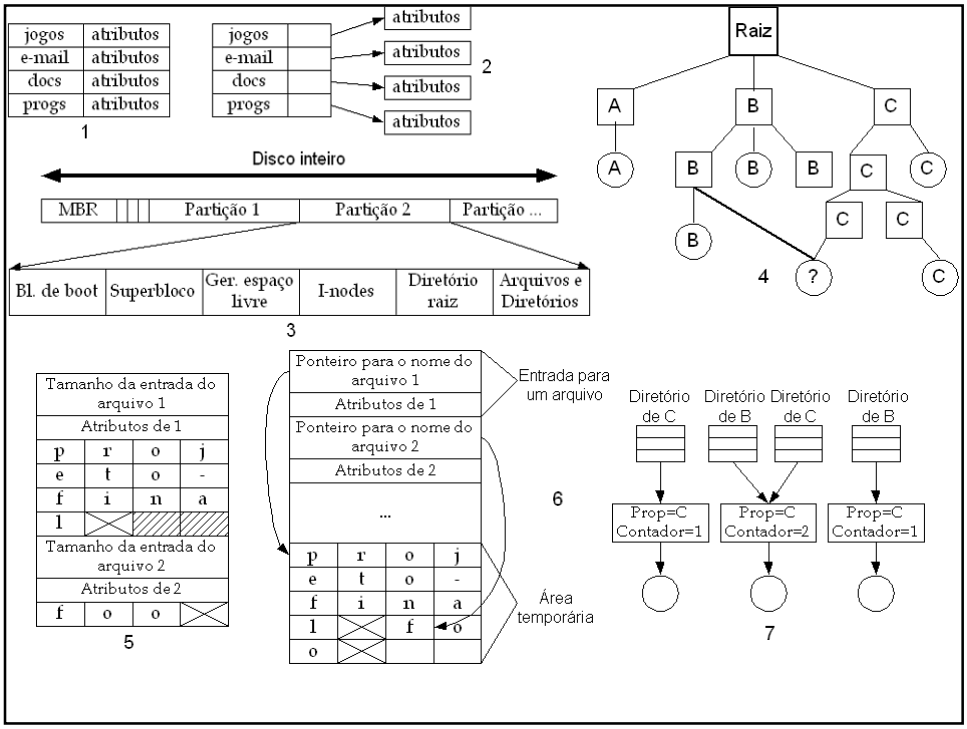
☑ **Não esquecer**: manutenção das informações em memória principal.

14

Esquema do Sistema de Arquivos

- Disco dividido em uma ou mais partições;
- Um SA para cada partição;
- Setor especial: setor 0, MBR ➔ registro de controle de inicialização do sistema;
  - O fim da MBR armazena a tabela de partição;
  - O BIOS localiza na MBR uma partição ativa e lê seu primeiro bloco (bloco de boot).
- **Compartilhamento**
  - Conveniente um mesmo arquivo aparecer em vários diretórios.
  - Conexão entre um arquivo e o diretório → **ligação**.
    - Grafo Acíclico Orientado.
  - Associação ao próprio arquivo;
  - Ligação simbólica.

15



Implementação de Arquivos

- Ponto chave:
  - Associar blocos de disco a arquivos.
- Alocação Contígua
  - mais simples de todos os esquemas de alocação;
  - cada arquivo é armazenado no disco como um bloco contínuo de dados.
  - armazenar um arquivo de 50 KB:
    - com blocos de 1 KB → 50 blocos consecutivos alocados;
    - com blocos de 2 KB → 25 blocos consecutivos alocados.
  - Vantagens:
    - simplicidade e desempenho.
  - Desvantagens:
    - conhecer o tamanho máximo do arquivo no momento de sua criação;
    - a fragmentação do disco.

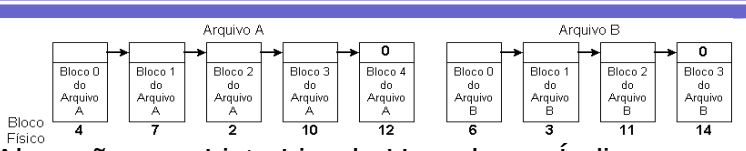
17

Implementação de Arquivos

- Alocação com Lista Ligada
  - manter o espaço em disco alocado ao arquivo com uma lista ligada de blocos.
  - em cada bloco além das informações do arquivo há um ponteiro para o próximo bloco.
  - vantagens:
    - não se perde com a fragmentação, pois qualquer bloco do disco pode ser usado;
    - a entrada do diretório só precisa armazenar o endereço em disco do primeiro bloco do arquivo.
  - Desvantagens:
    - acesso randômico extremamente lento;
    - necessidade de armazenar em cada bloco um ponteiro.

18

Implementação de Arquivos



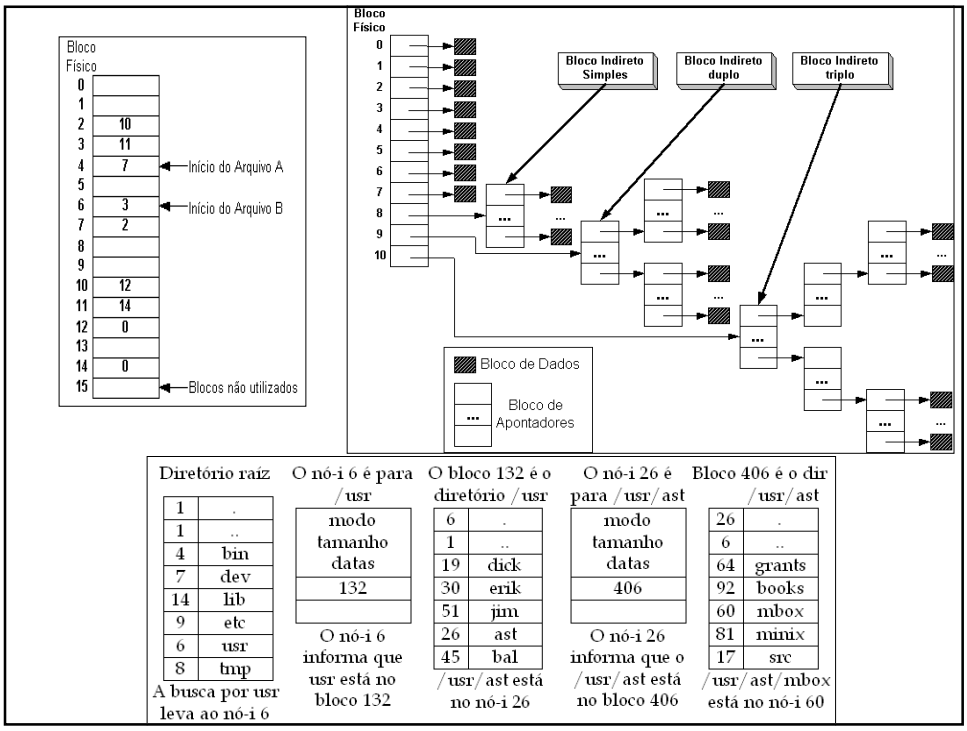
- Alocação com Lista Ligada Usando um Índice
  - ambas as desvantagens apontadas para a alocação com listas ligadas podem ser eliminadas tirando-se o ponteiro de cada um dos blocos e colocando-os em uma tabela ou índice na memória.
  - Vantagens:
    - todo o bloco fica disponível para armazenamento da informação;
    - apesar do acesso ser randômico, permite o acesso relativo. Toda a cadeia está na memória principal, podendo ser seguida sem necessidade de nenhum acesso a disco.
  - Desvantagem:
    - toda a tabela deve estar na memória durante todo o tempo.

19

Implementação de Arquivos

- Nós-i:
  - consiste em associar a cada arquivo uma pequena tabela denominada nó-i (nó-índice), que lista os atributos e os endereços em disco dos blocos do arquivo.
  - Primeiros endereços de disco são armazenados no próprio nó-i;
  - Em arquivos maiores, um dos endereços no nó-i é o endereço de um bloco de disco chamado bloco indireto simples.
  - bloco indireto duplo e blocos indiretos triplos relativos a este arquivo são usados se necessário.
  - Vantagens:
    - arquivos pequenos (que são a grande maioria) são acessados rapidamente, pois sua tabela de índices está na memória;
    - não há necessidade de ter toda a tabela de índices na memória principal.
  - Desvantagem:
    - arquivos maiores terão o acesso mais lento em função da necessidade de ler também os apontadores do disco.

20



Manutenção de Espaço Livre

1 - Tamanho do Bloco

- Leitura de um bloco k
  - posicionamento (seek): tempo para mover o braço do disco;
  - atraso rotacional (latência): tempo para que o setor gire até a cabeça de E/S;
  - taxa de transferência: Bytes transferidos em um s.
  - Geralmente medidos em milissegundos.
- Considerar a taxa de transferência: x MB/s
  - Ex.: seek (20 ms); latência (10 ms) taxa de transferência de 2 MB/s → (1 setor) = tempo de 0,2 ms.
  - Leitura de 1 KB:
    - Bloco do tamanho do setor (512 B) = 30,2 * 2 = 60,4 ms;
    - Bloco utilizando dois setores (1 KB) = 30,4 ms.

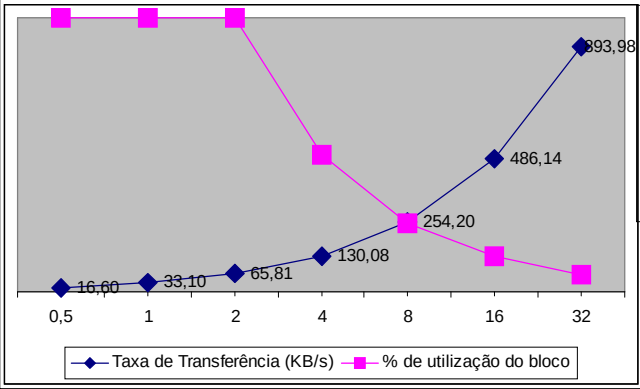
22

Manutenção de Espaço Livre

- Considerar o tempo de rotação
  - Tamanho da trilha X tempo de rotação
    - Blocos com k bytes: seek + latência + (k/cap.trilha)*TT

    Ex.: seek(20ms); latência(10ms); trilha(100 KB); TT(18ms);

    arq. de 2 KB para blocos (k) de 0,5; 1; 2; 4; 8; 16 e 32 KB.



- **Escalas**
- Taxa de transferência
  - 1 ~ 1000 (KB/s)
- % de utilização
  - 1 ~ 100 (%)

Manutenção de Espaço Livre

2 – Manutenção dos blocos livres

- Lista de espaço livre
  - Blocos que podem ser alocados

2.1 – Vetor de bits ou mapa de Bits

- Cada bloco é representado por um bit;
- Tamanho diretamente relacionado à capacidade do disco;
- Independente do estado do disco (cheio/vazio), o tamanho do mapa é fixo;
- Ex.: Disco de 10 GB com blocos de 1 KB
  - 10485760 KB → 10485760 bits
  - 1 bloco (1024B*8b) → 8192 bits → 1280 blocos (KB)

24

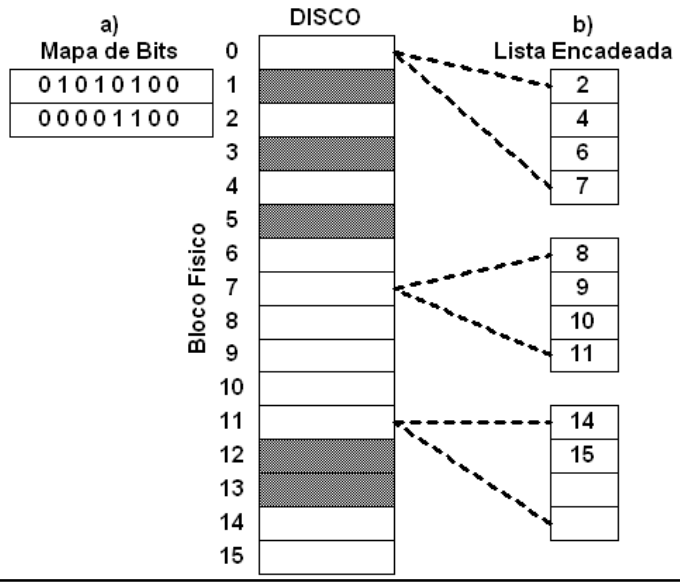
Manutenção de Espaço Livre

2.2 – Lista Encadeada

- Cada bloco guarda o máximo de blocos livres;
- Com blocos de 1 KB e números de bloco ocupando 32 bits temos:
  - 1 bloco (1024B * 8b) → 8192 bits → 256 blocos;
  - Uma entrada é ponteiro para próximo bloco com entradas livres (portanto 1 bloco armazena 255 blocos livres);
- Tamanho da lista depende da quantidade de blocos livres;
- Quanto mais ocupado estiver o disco, menor será a lista;
- Ex.: Disco de 10 GB com blocos de 1 KB
  - 10.485.760 KB (blocos) → 41.120,63 blocos
  - 41.121 blocos (KB) são necessários (40,12 MB)

25

Mapa de Bits e Lista Encadeada



Confiabilidade do SA

- Valor da informação superior ao do hardware.
- Necessidade de proteger o sistema de arquivos.
- Duas estratégias típicas:
  - Cópias de Segurança (*backups*);
  - Consistência do sistema de arquivos.
- Cópias de Segurança (1ª)
  - Possibilidade de trazer à normalidade o computador após a ocorrência de falhas com o sistema de arquivos; ou
  - restabelecer arquivos acidentalmente apagados.
  - ↓ cópia de segurança de todo o sistema mostra-se ineficiente (questões de tempo e quantidade de dados).
  - ↑ fazer cópia de arquivos específicos e que tenham sido modificados.

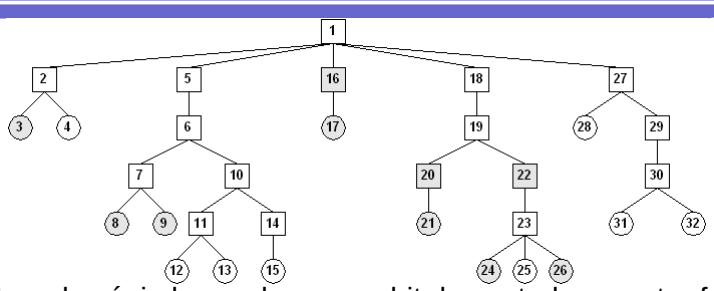
27

Confiabilidade do SA

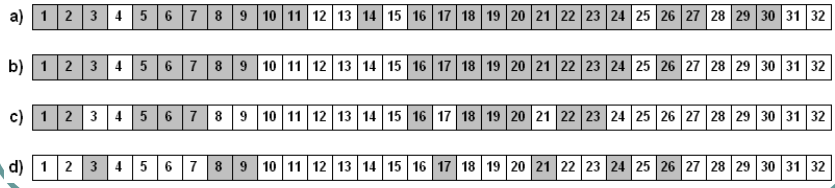
- Cópias incrementais
  - Fazer cópias completas periódicas; e
  - Fazer cópias diárias de arquivos modificados desde a última cópia completa.
- Duas alternativas de *backup* podem ser adotadas:
- Cópia Física (*dump físico*)
  - ↑ simplicidade e desempenho;
  - ↓ blocos defeituosos, restauração individual, cópia incremental.
- Cópia Lógica (*dump lógico*)
  - Cópia recursiva a partir de um diretório especificado.
  - Inclui subdiretórios, mesmo os inalterados.
  - ↑ Cópia incremental e restauração individual (esqueleto do AS).
  - ↓ Lista de blocos livres, links, arquivos especiais.

28

Backups – Estudo de Caso



Algoritmo de cópia baseado em um bit de controle e quatro fases.



29

Confiabilidade do SA

- Consistência do Sistema de Arquivos (2ª)
  - Estado inconsistente ocasionado por alterações em blocos que não foram efetivadas em disco.
  - Uso de utilitários que verificam a consistência do SA.

1 – Verificação da consistência por blocos

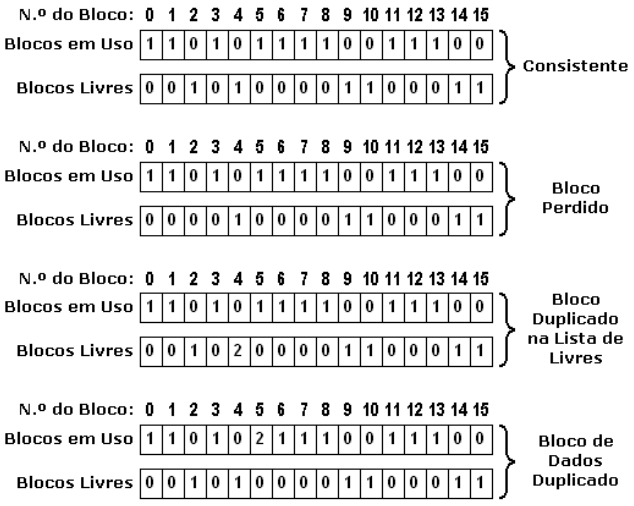
  - Uso de duas tabelas
  - freqüência do bloco em um arquivo;
  - freqüência do bloco na lista de livres (ou mapa de bits)
  - Em um sistema consistente, cada bloco estará em **uma** tabela.

2 – Verificação da consistência por arquivos

  - Inspeção recursiva de cada diretório a partir da raiz.
  - Comparação das entradas no diretório com os contadores de ligações de cada nó.

30

Consistência do Sistema de Arquivos



Desempenho do Sistema de Arquivos

- Acesso a disco é muito mais lento que o acesso a memória.
- Implementação de SA com otimizações.
- Três otimizações principais:
  - Cache de blocos ou cache de buffer;
  - Leitura antecipada de blocos;
  - Redução do movimento do braço do disco.

1 – Cache de blocos

- Coleção de blocos (logicamente de disco) mantidos em memória;
- Uso de algoritmo com comportamento típico;
- Algoritmo alternativo: Menos Recentemente Usado.

Desempenho – cache (cont.)

- Problema com blocos críticos;
- Solução: *uso do MRU modificado*.
  - Considera dois fatores:
  - Probabilidade do bloco ser novamente necessário; e
  - Se o bloco é essencial para a consistência do sistema.

➲ Dividir os blocos em categorias.

☑ Não manter dados na cache por muito tempo.

- Unix:
  - Uso de chamada de sistema especial (*sync*);
  - Projeto prioriza o SO baseado em discos rígidos.
- MS-DOS:
  - *Write-through cache*: blocos modificados são imediatamente escritos (projeto baseado em discos flexíveis).

33

Desempenho do Sistema de Arquivos

2 – Leitura Antecipada dos Blocos

- Tentar transferir blocos para a cache antes que eles sejam necessários.
  - Boa estratégia para arquivos lidos seqüencialmente.
  - Péssima idéia para acessos randômicos.
- Alternativa:
  - Uso de um bit de controle para as duas situações:

1 → modo de acesso seqüencial;

0 → modo de acesso aleatório.

  - Inicialmente o arquivo é setado com **1**.
  - Em caso de posicionamento, o bit vira 0.
  - Se recomeçarem leituras seqüenciais, o bit volta para 1.

34

Desempenho do Sistema de Arquivos

3 – Redução do movimento do braço do disco

- Redução na quantidade de movimentos do disco colocando blocos com mais acessos em seqüência.

1 - Ao criar arquivos, reservar mais blocos que os usados:

- (Redução do número de posicionamentos).

2 – Ou alocar blocos consecutivos em um mesmo cilindro:

- (Redução do atraso rotacional).

- Manipulação de arquivos geralmente necessitam acessos duplos ao disco:
  - Um acesso para a localização do índice o outro para o bloco.
  - Alocação dos índices no centro do disco**; ou** agrupar os índices aos blocos de dados.

35

Exemplos de Sistemas de Arquivos

- **ISO 9660**
  - Padrão internacional de 1988 para CD-ROMs;
  - Possibilita que o CD-ROM seja lido por qualquer computador, independentemente do SO;
  - Impõe certas restrições para manter compatibilidade;
  - Blocos mantidos em uma única espiral contínua;
  - Blocos lógicos com 2352 bytes (*payload*: 2 KB);
  - Fator de conversão: 1 s → 75 blocos;
  - Conjunto de blocos livres e restritos: **manter compatibilidade**;
  - Algumas informações em ASCII e outras em binário;
  - Especificado em três níveis: quanto mais baixo o nível, mas restritiva é a definição;
  - Extensões: *Rock Ridge* e *Joliet*.

36

Exemplos de Sistemas de Arquivos

- **CP/M**
  - Sistema operacional dos primeiros microcomputadores;
  - Extremamente compacto: para máquinas com 64 KB;
  - Um nível de diretório com entradas de tamanho fixo;
  - Uso de mapa de bits em memória, sem gastar disco;
- **FAT**
  - Versão com mais recursos do CP/M;
  - O sistema de arquivos forma uma árvore a partir do raiz;
  - Nome com 8+3 caracteres com bits para atributos;
  - Três versões de sistema de arquivos: FAT 12, FAT 16 e FAT 32
  - As vantagens do FAT 32 implicam não somente o tamanho da partição, mas também o tamanho do cluster.
  - Ex.: HD de 2 GB com blocos de 4 KB=512K blocos (2MB RAM).

37

Exemplos de Sistemas de Arquivos

- **NTFS**
  - desenvolvido especificamente para o NT;
  - mantém compatibilidade com FAT 16 e FAT 32;
  - endereçamento de 64 bits;
  - um setor pode ser diretamente endereçado;
  - recursos de segurança e de gerenciamento de disco (ex.: *hotfix*).
- **EXT**
  - usado na maioria das distribuições Linux;
  - suporta partições de 4 TB e nome de arquivos de 255 caracteres;
  - EXT 3: versão atual, com melhoria no suporte a tolerância a falhas;
  - Manutenção de um “diário” das operações realizadas.

38