TÓPICO 7

Gerência de Arquivos

Sistemas de Arquivos

- Em um computador, os dados podem ser armazenados em vários dispositivos físicos diferentes (disco flexível, fita, disco rígido, CD, etc).
- Para simplificar o tratamento, o SO fornece uma visão lógica e uniforme do sistema de armazenamento.
- O SO define também uma unidade lógica de armazenamento: o arquivo.
- Os arquivos são entidades lógicas mapeadas em dispositivos físicos.

Sistemas de Arquivos

- Definição: um arquivo é uma coleção de dados relacionados entre si.
- Cada arquivo possui um nome, que o identifica. Além do nome, o arquivo possui outros atributos tais como tipo, nome do criador, tamanho, etc.
- As informações contidas em um arquivo são *persistentes*. Elas são armazenadas em dispositivos não-voláteis.
- O sistema de arquivos é o módulo do SO responsável pela criação da abstração de arquivo e por seu gerenciamento.

Identificação do Arquivo

- Para que um arquivo seja referenciado, um nome deve ser atribuído a ele.
- A atribuição de nomes a arquivos é feita no momento de sua criação.
- As regras a serem adotadas para a confecção do nome de um arquivo variam de acordo com o SO.

Identificação do Arquivo

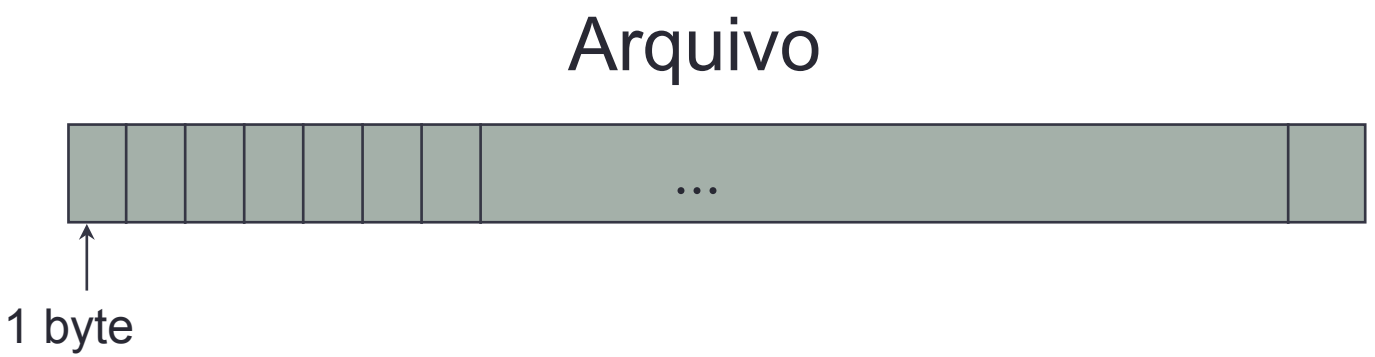
- Exemplos:
  - DOS: <nome>.<extensao>, onde o <nome> pode conter até 8 caracteres e a extensão contem até 3 caracteres. Não sensitivo ao caso (case-insensitive).
  - UNIX:<nome>, onde <nome> pode conter até 255 caracteres, dependendo do sistema de arquivos utilizado. Sensitivo ao caso (case-sensitive).

Estruturação do Arquivo

- O servidor de arquivos deve implementar a abstração de arquivo para o restante do sistema. Para tanto, ele deve determinar como o arquivo será estruturado internamente.
- As estruturas de arquivos mais comuns são:
  - sequência de bytes
  - sequência de registros
  - árvore de registros

Estruturação do Arquivo
Seqüência de Bytes

- Neste caso, não há estrutura. O arquivo é simplesmente enxergado como uma sequência de bytes.
- O SO não tem conhecimento do significado e da estruturação dos campos que compõem um arquivo.



Estruturação do Arquivo
Seqüência de Bytes

- Vantagens: flexibilidade, possibilidade de uso não convencional de arquivos
- Desvantagens: em alguns casos, acesso não otimizado aos dados
- Exemplos: MS/DOS e Unix

Estruturação do Arquivo
Seqüência de Registros

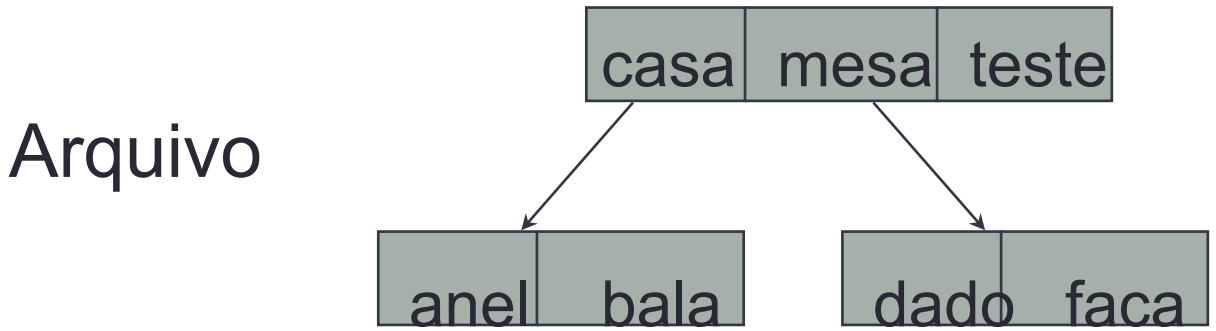
- Neste caso, o arquivo é estruturado pelo SO como uma sequência de registros de tamanho fixo.
- As operações de leitura e escrita retornam um registro.

Estruturação do Arquivo
Seqüência de Registros

- Este tipo de estruturação foi muito popular no tempo dos cartões perfurados e das impressoras com número fixo de caracteres.
- Vantagens: acesso otimizado por registro
- Desvantagens: dificuldade em se determinar o tamanho do registro, perda de flexibilidade, espaços não utilizados.
- Exemplo: CP/M

Estruturação do Arquivo
Árvore de Registros

- Neste caso, o arquivo é uma árvore de registros de tamanho variável. Cada registro é composto por um campo chave, contido em uma posição fixa. A árvore é ordenada pelo campo chave.
- Esta estruturação visa otimizar o acesso randômico a registros.



Estruturação do Arquivo
Árvore de Registros

- Vantagens: acesso randômico otimizado
- Desvantagens: complexidade da implementação, estrutura fixa de armazenamento
- Exemplos: sistemas de grande porte, MUMPS

Tipos de Arquivos

- Cada sistema de arquivos determina os tipos de arquivos suportados por ele.
- Sistemas como o Unix e o DOS suportam os seguintes tipos de arquivos:
  - *arquivos regulares*: contem os dados do usuário
  - *arquivos diretório*: arquivos utilizados na manutenção do sistema de arquivos.
  - *arquivos especiais*: arquivos ligados a dispositivos de E/S.

Métodos de Acesso ao Arquivo

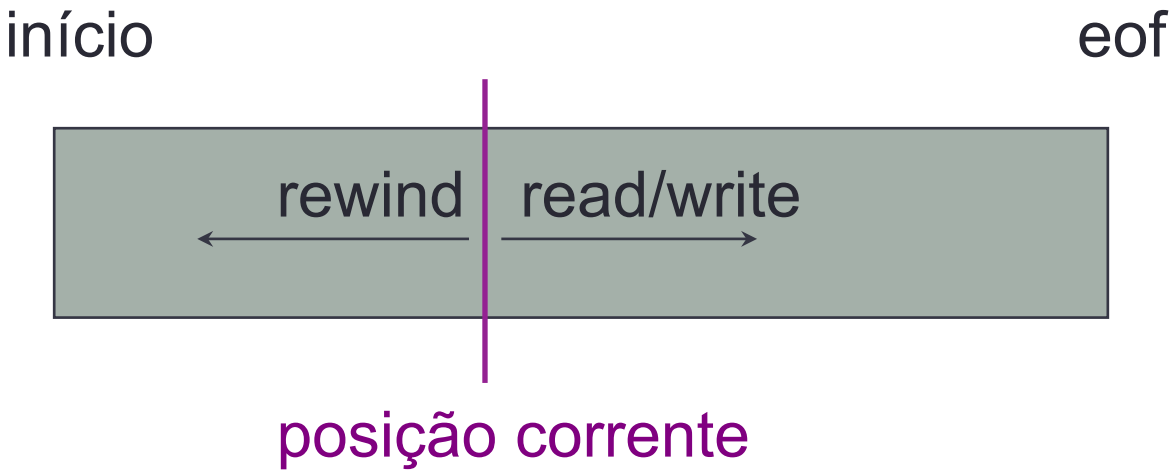
- Os métodos de acesso determinam como os dados contidos no arquivo serão recuperados pelo usuário.
  - Acesso seqüencial
  - Acesso direto (ou aleatório)

Acesso Seqüencial

- Este método de acesso utiliza basicamente as operações de leitura e escrita.
  - Operação de leitura (read next): lê a *próxima* posição do arquivo e avança o ponteiro
  - Operação de escrita (write next): coloca o dado no final do arquivo e avança o end-of-file (EOF).

Acesso Seqüencial

- Este método é baseado no funcionamento da fita magnética.
- Os arquivos sequenciais podem ser rebobinados e lidos inúmeras vezes.



Acesso Direto ou Aleatório

- Este método de acesso utiliza basicamente as operações de leitura e escrita.
  - Operação de leitura (read n): lê a posição do arquivo apontada por n. Geralmente, n é o número do bloco.
  - Operação de escrita (write n): coloca o dado no bloco n.
- Este método é baseado no funcionamento do disco.

Acesso Direto ou Aleatório

- Os sistemas de arquivos podem oferecer simultaneamente os dois métodos de acesso e o usuário determina o método a ser utilizado na criação do arquivo.
- A maioria dos SO modernos oferece somente o método de acesso direto.

Atributos do Arquivo

- Além do nome e dos dados, o SO associa a cada arquivo um conjunto de informações que auxiliam na gerência dos mesmos.
- Estas informações adicionais são chamadas atributos e estão geralmente contidas na tabela de arquivos.
- São exemplos de atributos de arquivos:
  - *proteção*: indica as permissões de acesso ao arquivo
  - *password*: password necessária para o acesso ao arquivo

Atributos do Arquivo

- *criador:* usuário criador do arquivo
- *owner*: proprietário atual do arquivo
- *flag de oculto*: impede que o nome do arquivo apareça na lista de arquivos
- *flag de temporário*: o arquivo é deletado quando o processo que o criou morrer.
- *tamanho*: tamanho atual do arquivo
- *tamanho máximo*: tamanho máximo que o arquivo pode atingir.
- *instante da última modificação e da criação*

Operações sobre o Arquivo

- As operações sobre arquivos geralmente levam em consideração um conjunto de entidades de um sistema de arquivos.
- São elas a tabela de arquivos, o diretório e a posição corrente.
  - tabela de arquivos: possui uma entrada para cada arquivo. Contem as informações referentes aos arquivos.

Operações sobre o Arquivo

- diretório: caminho que leva ao arquivo. Ao ser criado, o arquivo é posicionado em um diretório e a tabela de arquivos deste diretório deve conter uma entrada para o novo arquivo.
- posição corrente do arquivo: ponteiro que diz qual a posição do arquivo será acessada na próxima vez. As operações de leitura e escrita incrementam o valor da posição corrente do arquivo do número de bytes lidos/escritos.

Operações sobre o Arquivo

1) ***CREATE:*** cria um arquivo. Deve-se criar uma entrada para o novo arquivo no diretório especificado. Além disso, deve-se criar uma entrada na tabela de arquivoscom o nome do criador, data de criação e permissões de acesso. O arquivo recem-criado não possui dado nenhum.

2) ***DELETE:*** Libera o espaço ocupado pelo arquivo, deleta a entrada do diretório que aponta para o arquivo e deleta a entrada na tabela de arquivos.

Operações sobre o Arquivo

3) ***OPEN:*** Antes de ser utilizado, o arquivo precisa ser aberto. Essa chamada cria um mapeamento entre a entrada da tabela de arquivos referente ao arquivo a ser aberto para a tabela de processos do processo que executou o OPEN. Isso é feito através de uma tabela de descritores de arquivos. Este comando retorna um *descritor de arquivo,* que será utilizado em todas as operações subsequentes sobre o arquivo.

4) ***CLOSE:*** Desfaz o mapeamento entre a entrada da tabela de arquivos referente ao arquivo da entrada da tabela de processos do processo. Todos os acessos subsequentes ao descritor de arquivo serão inválidos.

Operações sobre o Arquivo

5) ***READ:*** para ler um arquivo, o processo deve especificar um descritor válido, a quantidade de dados a serem lidos e um local (buffer) em memória onde os dados lidos devem ser colocados. A maioria das operações de leitura lê a partir da posição corrente.

6) ***WRITE:*** para escrever dados em um arquivo, o processo deve especificar um descritor válido, um buffer local que contenha os dados a serem escritos e o tamanho destes dados. Normalmente, os dados são escritos em um arquivo a partir da posição corrente.

Operações sobre o Arquivo

7) ***APPEND:*** escreve dados em um arquivo, a partir do final do mesmo.

8) ***SEEK:*** utilizado para arquivos de acesso randômico. Para se fazer um seek é necessário um ponteiro para o arquivo. O SEEK faz com que o ponteiro fornecido seja a nova posição corrente do arquivo.

9) ***RENAME:*** muda o nome do arquivo. Simplesmente altera o nome do arquivo na tabela de arquivos.

Manipulação de Arquivos

- Esquema geral de manipulação de arquivos
  1. Criar o arquivo
  2. Abrir o arquivo
  3. Ler/Escrever/Append/seek
  4. Fechar o arquivo
  5. Deletar o arquivo (ou não)

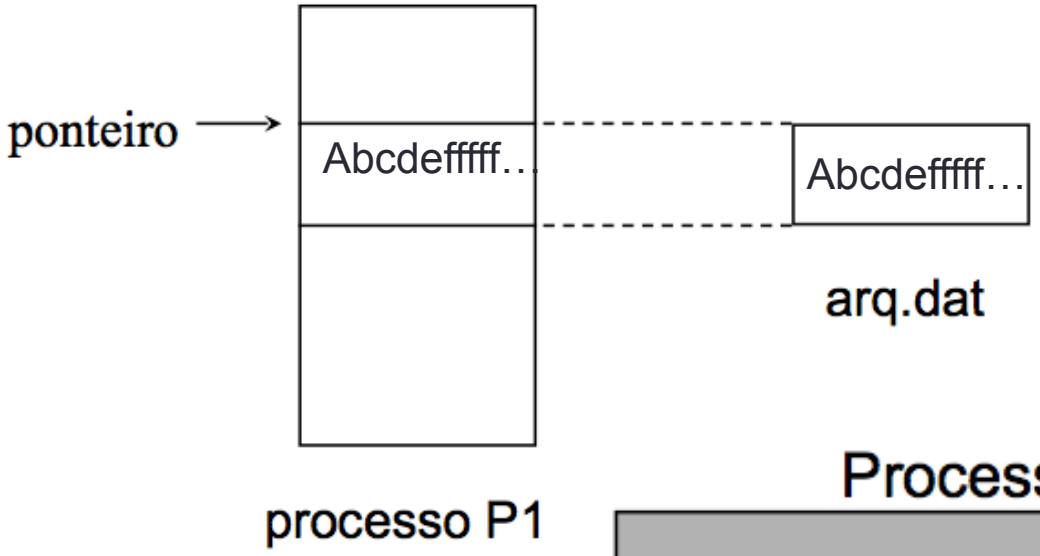
Arquivos Mapeados na Memória

- A manipulação de arquivos da maneira tradicional é extremamente complexa, comparada as operacões de acesso à memória.
- A idéia envolvida em sistemas de arquivos mapeados em memória é colocá-los (através de uma operação de mapeamento) no espaço de endereçamento do processo.
- Assim, a manipulação de dados, seja em memória seja em arquivos, ocorre de maneira uniforme.

Arquivos Mapeados na Memória

- O mapeamento e o “desmapeamento” de arquivos é realizado através de operações MAP e UNMAP.
- A operação MAP é a equivalente ao OPEN. Ao invés do descritor de arquivos, ela retorna um ponteiro para o início do arquivo.
- As operações de leitura e escrita de dados são feitas simplesmente através de acessos à memória (LOAD e
- STORE) em relação ao ponteiro retornado.

Arquivos Mapeados na Memória



Processo P1

```
            ...
ponteiro = map("arq.dat");
a = *ponteiro;
*(ponteiro +10) = 2;
unmap(ponteiro);
            ...
```

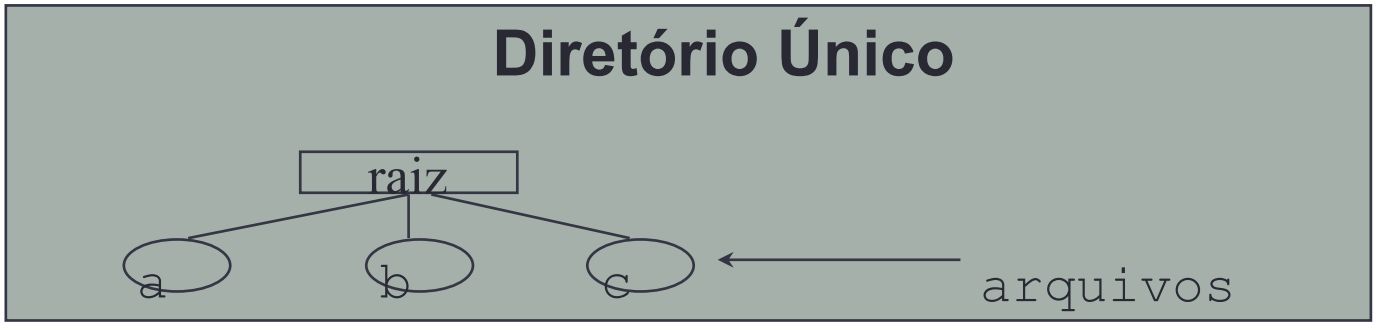
Arquivos Mapeados na Memória Vantagens/Desvantagens

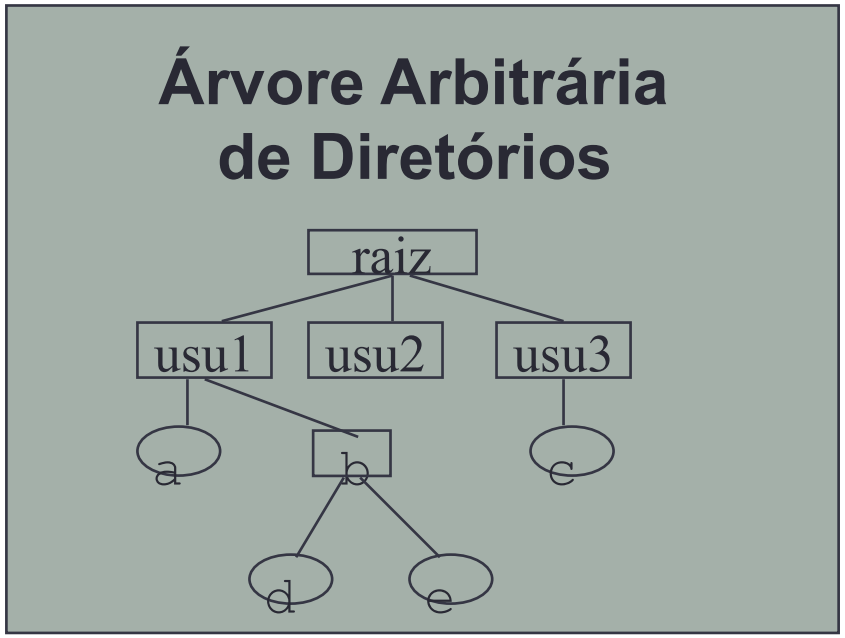
- Vantagens:
  - maior simplicidade e uniformidade da programação
- Desvantagens:
  - Em um sistema paginado, o fim de arquivo deve ser marcado por um caracter especial (EOF)
  - Problemas de inconsistência
  - Mapeamento de arquivos cujo tamanho é maior do que o espaço de endereçamento.

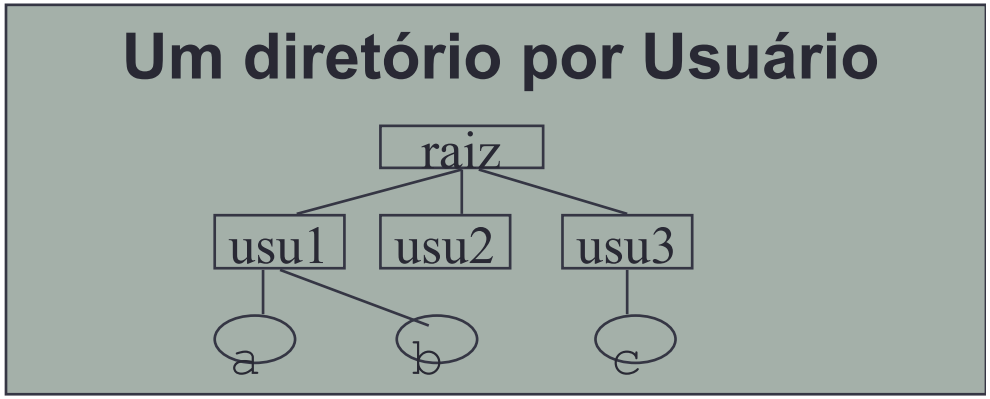
Diretórios

- Os diretórios são utilizados para auxiliar a disposição dos arquivos no sistema de arquivos.
- Os diretórios são estruturas hierárquicas, com uma entrada por arquivo. Cada entrada guarda o nome do arquivo, seus atributos e os endereços do(s) disco(s) onde ele está armazenado.

Organização dos Diretórios

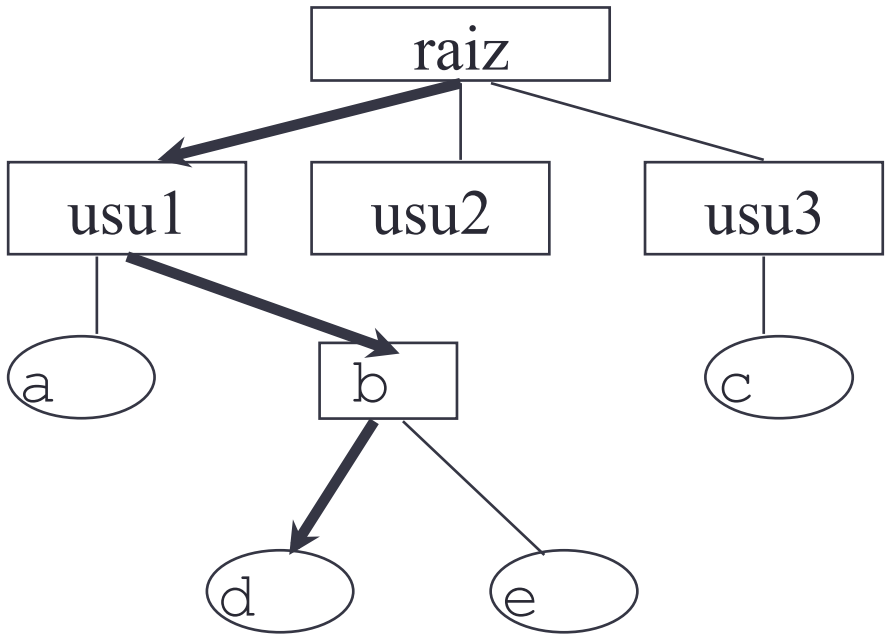






Nomes de Arquivos

- Como os arquivos estão organizados em diretórios, o "caminho" que é necessário se percorrer para se chegar até o arquivo também faz parte do seu nome.



Nomes de Arquivos

- Nome absoluto: nome que vai do diretório raiz até o arquivo.
  - Ex: /usu1/b/d
- Nome relativo: nome relativo ao diretório corrente.
  - *Diretório corrente*: diretório no qual estamos posicionados atualmente.
  - Ex: (diretório corrente=usu1) b/d, ../usu3/c
- Diretório home: diretório default do usuário

Operações sobre Diretórios

- CREATEDIR: cria um diretório. Cria uma entrada para o novo diretório no diretório corrente (ou no diretório especificado).
- REMOVEDIR: remove a entrada referente ao diretório no diretório corrente (ou no diretório especificado). A priori, somente diretórios vazios são removidos.
- RENAMEDIR: muda o nome do diretório.
- LINK: cria um novo caminho para um arquivo já existente.
- UNLINK: remove um caminho para um arquivo.

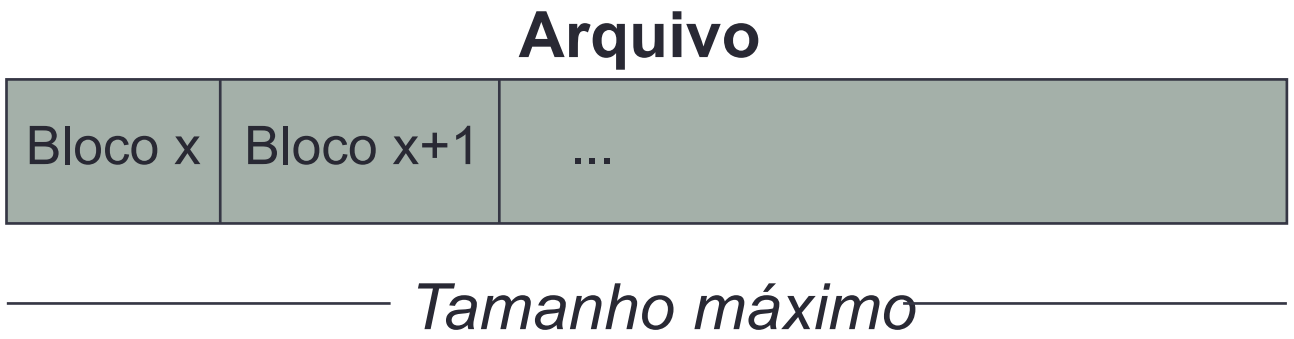
Implementação de Arquivos

- A primeira coisa a se decidir na implementação de arquivos é como os mesmos serão armazenados. Os dados contidos nos arquivos são armazenados em blocos de disco. Para a associação de blocos de disco a arquivos, temos as seguintes abordagens:
  - Alocação contígua
  - Alocação com lista encadeada e índice
  - Nós-i

Implementação de Arquivos
Alocação Contígua

- Na criação do arquivo é reservado um espaço contíguo em disco.
- Para se guardar os endereços de disco associado a um arquivo, é extremamente simples: só necessitamos do endereço inicial do arquivo no disco e de seu tamanho.
- A recuperação de dados é extremamente rápida pois, como os blocos são contíguos, o movimento do braço do disco é mínimo.

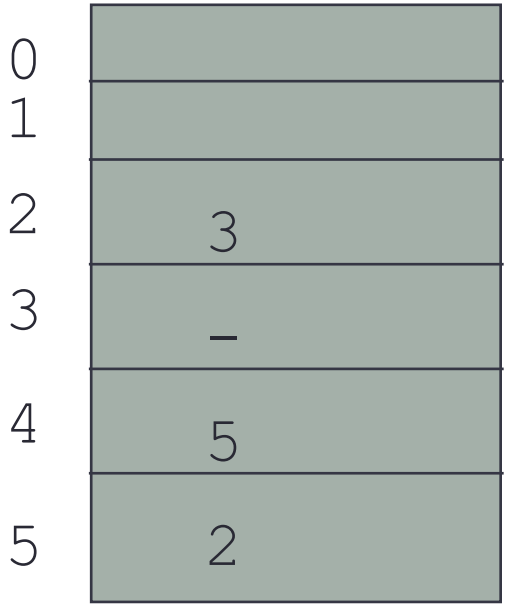
Implementação de Arquivos
Alocação Contígua



- Desvantagens:
  - É necessário conhecer o tamanho máximo do arquivo na sua criação.
  - Fragmentação do disco

Implementação de Arquivos
Lista Encadeada

- Neste caso, continuamos a manter uma lista encadeada de blocos. Na implementação, no entanto, o ponteiro é colocado em uma tabela geral de blocos.



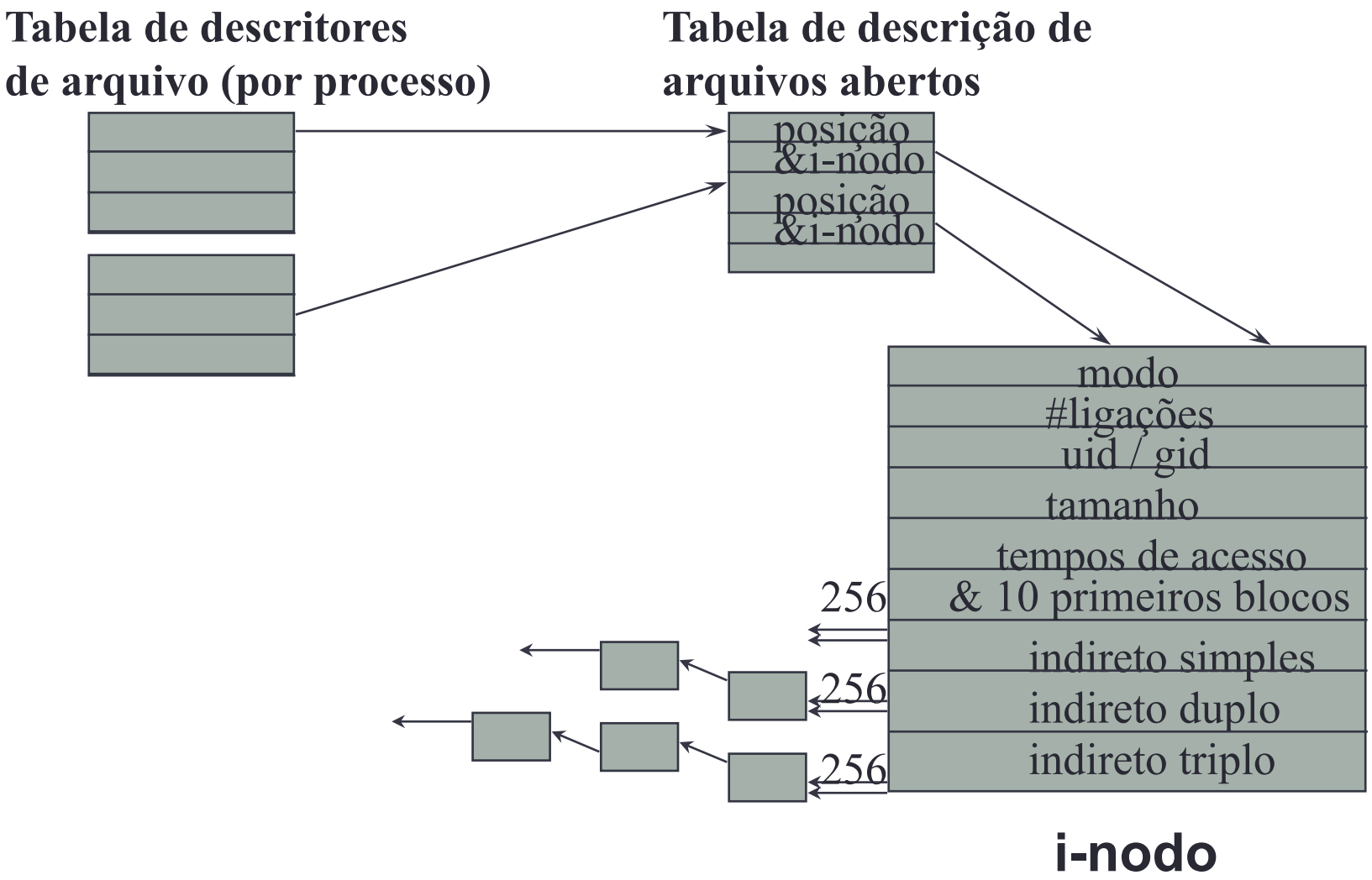
Implementação de Arquivos
Lista Encadeada

- O acesso randômico possui uma implementação relativamente simples.
- Desvantagem: como esta tabela é única para todos os blocos físicos do disco, ela necessita ficar em memória o tempo todo. Devido ao seu tamanho, ela geralmente é paginada.

Implementação de Arquivos
I-nodos

- Neste método, cada arquivo possui uma pequena tabela denominada tabela de i-nodos, que contem os atributos e os endereços dos blocos físicos alocados ao arquivo.
- Os primeiros endereços de disco são armazenados no próprio i-nodo, que é transferido do disco para a memória no momento da abertura do arquivo.
- Em arquivos maiores, o i-nodo contem o endereço de um bloco de disco denominado *bloco indireto simples*. Este bloco é simplesmente uma tabela de endereços de blocos de disco. Podemos ter também indireção em segundo e terceiro níveis.

Implementação de Arquivos
I-nodos



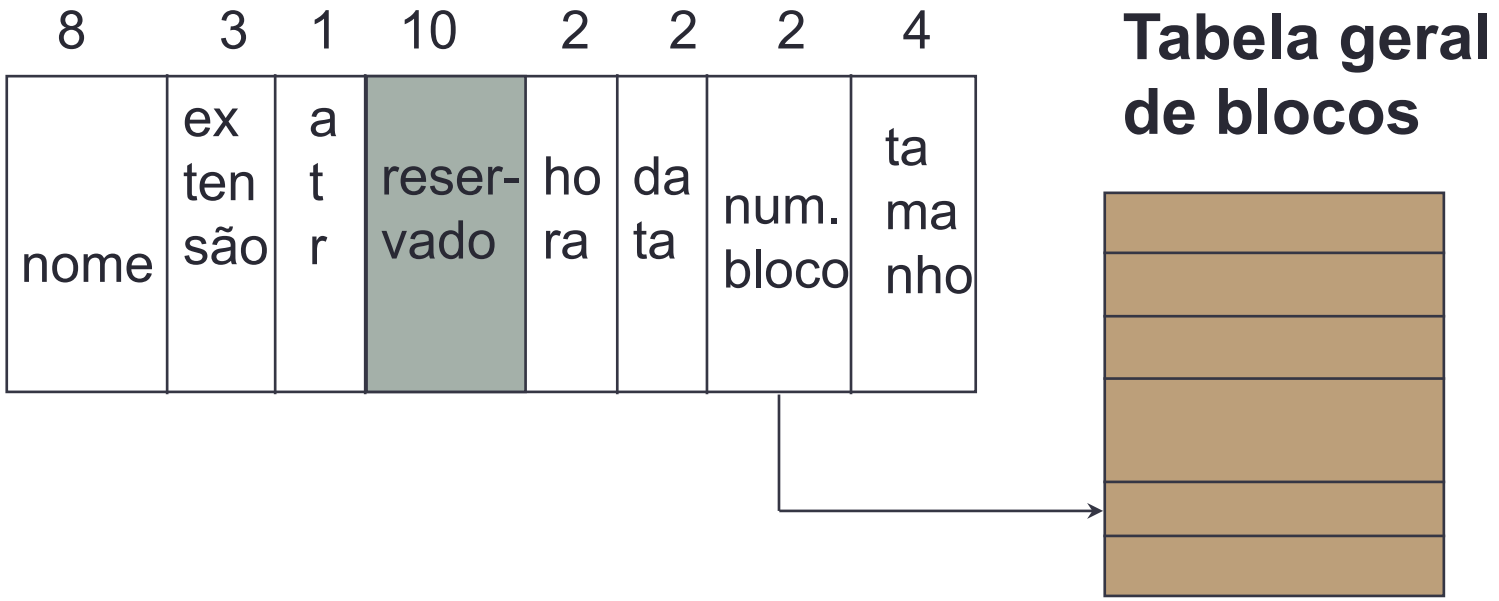
Implementação de Arquivos
Exemplo 1 -CP/M

- Organização em diretório único: todos os arquivos estão imediatamente abaixo do diretório raiz.
- Os números dos blocos de disco e os atributos são armazenados na própria entrada do diretório.
- Se um arquivo utilizar mais blocos que o máximo de blocos contidos em uma entrada, uma nova entrada é alocada para o mesmo arquivo.
- Entrada do diretório:

1	8	3	1	1	16
cod usu	nome	ex ten são	con tinua	n. blocos	números dos blocos

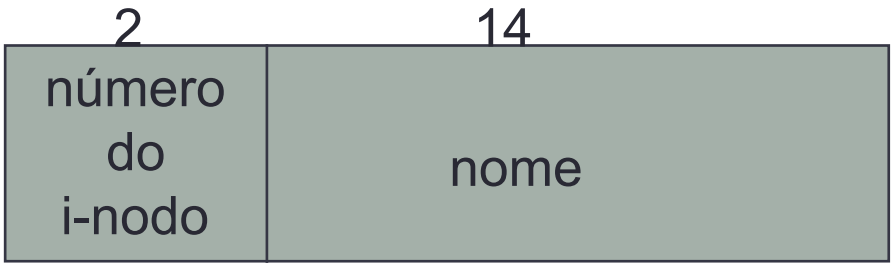
Implementação de Arquivos
Exemplo 2 - DOS

- Organização em árvore de diretórios
- Entrada do diretório:



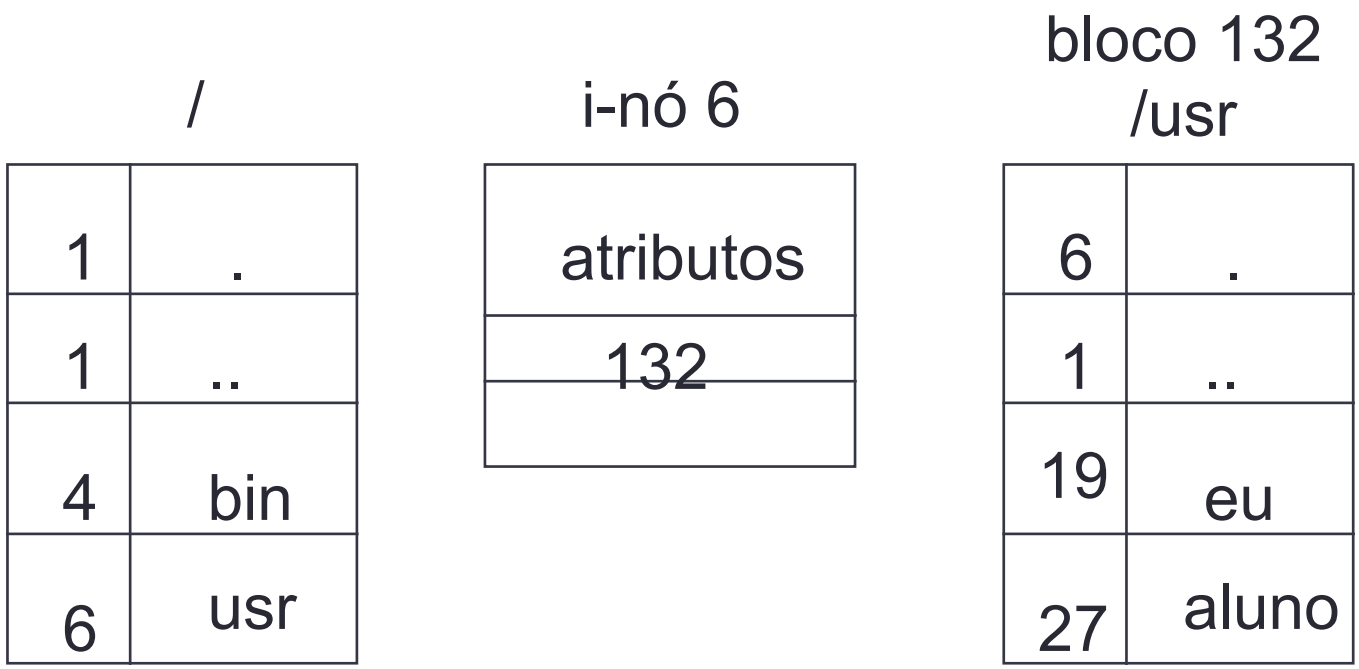
Implementação de Arquivos
Exemplo 3 - Unix

- Organização dos diretórios em árvore
- Implementação por i-nodos
- Entrada do diretório:



Implementação de Arquivos
Exemplo 3 - Unix

- Exemplo: /usr/aluno



Tamanho do Bloco

- Os arquivos são normalmente armazenados em blocos de tamanho fixo.
- Os blocos são unidades lógicas, tratadas pelo sistema de arquivos. Estas unidades devem ser mapeadas no disco físico.
- O disco físico é geralmente organizado em cilindro, trilha e setor. Assim, os tamanhos destas entidades são candidatos óbvios para o tamanho do bloco (unidade de alocação).

Tamanho do Bloco

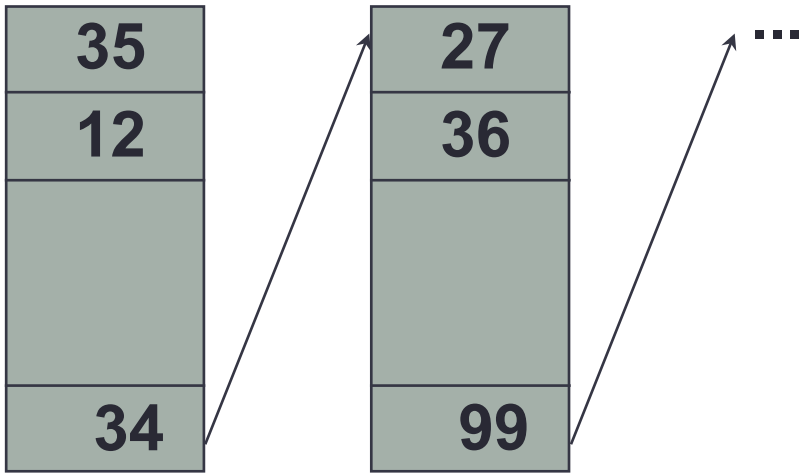
- Unidades de alocação grandes (cilindro):
  - D: maior fragmentação interna: grande parte do último bloco do arquivo é perdida
  - V: tabela de blocos menor
- Unidades de alocação pequenas (setor):
  - V: menor fragmentação interna;
  - D: tabela de blocos grande
- O usual é optar por um tamanho de bloco entre 512bytes e 2k, geralmente correspondente ao tamanho do setor.

Gerência de Espaço em Disco

- O Sistema de Arquivos deve manter o controle dos blocos livres em disco para fazer a alocação.
- Para isso, existem basicamente duas abordagens:
  - Lista encadeada
  - Mapa de bits

Gerência de Espaço em Disco
Lista Encadeada

- A lista encadeada que controla os blocos livres é disposta ela mesma em blocos.
- A lista encadeada guarda somente os blocos livres.



Gerência de Espaço em Disco
Lista Encadeada

- Como vantagem, podemos citar a alocação rápida de blocos.
- Porém, este esquema:
  - consome muito espaço com as listas
  - há necessidade de operações de lista

Gerência de Espaço em Disco
Mapa de Bits

- Neste caso, utilizamos 1 bit para cada bloco do disco. Blocos livres são representados por 1 e blocos alocados são representados por 0.

0000000000110010
0111111000000111
0110011000000100

Gerência de Espaço em Disco
Mapa de Bits

- Como vantagens deste esquema, podemos citar o pouco espaço consumido. Além disso, o espaço consumido é fixo.
- A desvantagem é a possibilidade de operações demoradas de alocação de blocos.

Confiabilidade do Sistema de Arquivos

- A perda de dados ou mesmo de arquivos inteiros é extremamente onerosa para o usuário. No mínimo, ele vai gastar muito tempo para colocar o SA no estado que estava antes da perda.
- Existem algumas técnicas que podem ajudar a reduzir a probabilidade da perda total da informação:
  - Gerência de blocos ruins
  - Backups
  - Gerência da consistência

Confiabilidade do Sistema de Arquivos

- Os discos rígidos geralmente saem da fábrica com um conjunto de blocos defeituosos.
- Para tratar os blocos ruins, existem duas abordagens:
  - Por Hardware
  - Por Software

Gerência de Blocos Ruins

- Por hardware: Um setor específico do disco arqmazena a lista de blocos ruins. Quando a controladora de disco é ativada pela primeirra vez, ela faz um mapeamento dos blocos ruins em “blocos reservas”. Desta maneira, os blocos ruins nunca serão utilizados.
  - A capacidade do disco é reduzida do número de blocos ruins.

Gerência de Blocos Ruins

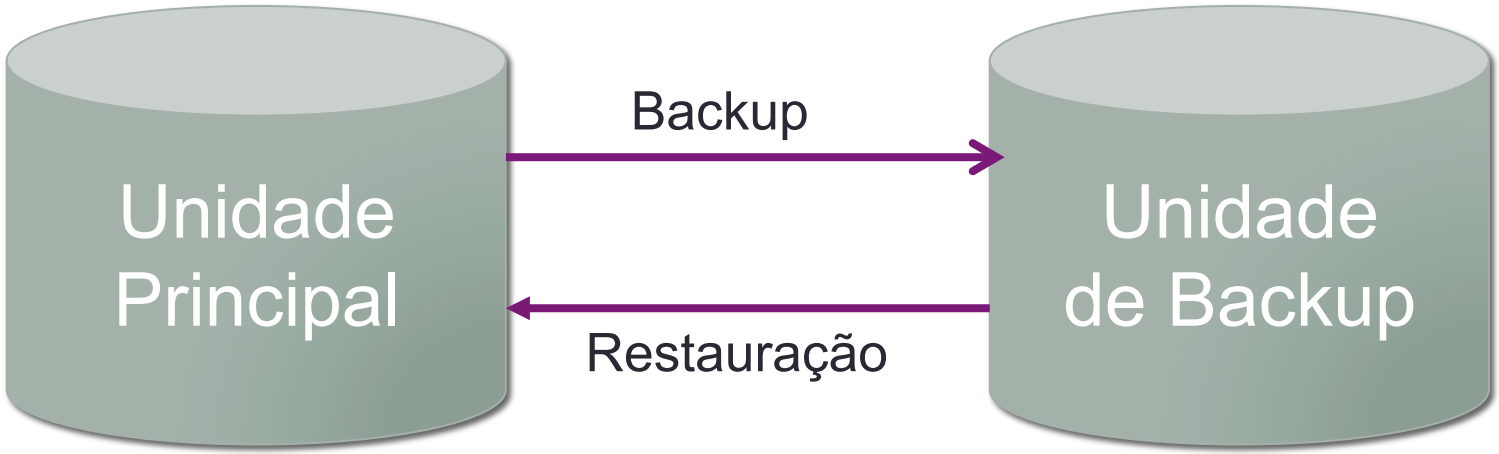
- Por software: É construído um arquivo contendo todos os blocos defeituosos. Como os blocos fazem agora parte de um arquivo, o SA não os coloca na lista de blocos livres.
  - O SA tem que impedir que este arquivo seja lido ou escrito.

Backups

- Mesmo com todos os controles de defeitos, existe sempre a possibilidade real de perda irreparável da informação (crash do disco).
- Para tratar este problema, a única solução é copiar periodicamente as informações do disco para um outro lugar (disco ou fita), duplicando-as (ou triplicando-as).
- A técnica mais antiga consiste em copiar as informações do disco para a fita magnética.

Backup e Restauração

- Backup: cópia periódica dos arquivos para outra unidade de armazenamento
  - Outro disco, storage, pen drive, DVD, etc.
- Restauração: cópia dos arquivos da unidade de backup para a unidade principal.



Backups

- Existem vários tipos de backup, dentre eles:
  - *Backup total*: faz a cópia de todos os arquivos do disco
  - *Backup incremental*: faz a cópia somente dos arquivos que foram alterados desde o último backup total ou desde o último backup incremental

Backups

- Em uma instalação, é comum se fazer um backup total por semana (e.g. sexta-feira) e backups incrementais diários. Assim, mantemos no máximo 7 backups: 1 completo e 6 incrementais.
- A recuperação de um backup nesse esquema é a seguinte (admita que hoje é segunda-feira):
  - Recupera o backup total
  - Aplica o backup de Sábado
  - Aplica o backup de Domingo

Consistência do SA

- O SA manipula dados e também ponteiros para estes dados (diretórios, i-nodos, etc). Uma inconsistência nesses ponteiros pode causar sérios danos no sistema de arquivos.
- Geralmente, todo sistema de arquivos possui um processo chamado “verificador de arquivos” que roda na inicialização do SA geralmente após uma falha.
- São feitos basicamente dois tipos de verificação: em bloco ou em arquivos.

Consistência do SA
Verificação em Blocos

- É construída uma tabela com dois contadores por bloco:
  - contador de num. de vezes de utilização
  - contador de num. de vezes de presença na lista de blocos livres.

Consistência do SA
Verificação em Blocos

- O procedimento é o seguinte:
  - Lê-se todos os i-nodos. Cada vez que um número de bloco for lido, seu contador de utilização é incrementado.
  - Lê-se a lista de blocos livres. Cada vez que um número de bloco for lido, seu contador de presença nos blocos lidos é incrementado.
- O SA está consistente se todo bloco possuir o valor 1 no primeiro ou no segundo contador. Caso contrário, o SA está inconsistente.

Consistência do SA
Verificação em Blocos

- *Inconstência 1 - Bloco perdido*
  - Existem blocos com os dois contadores zerados.

BL	**1000100100011111**
BU	**0111001011110000**

  - Neste caso, o bloco é simplesmente adicionado à lista de blocos livres

Consistência do SA
Verificação em Blocos

- *Inconsistência 2* - Duplicação do bloco livre
  - Existem blocos que aparecem duas ou mais vezes na lista de blocos livres.

BL **1000120100011111**

BU **0111001011110000**

  - Neste caso, retira-se a duplicação da lista de blocos livres

Consistência do SA
Verificação em Blocos

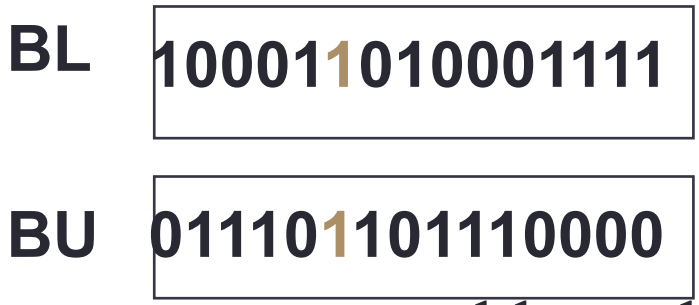
- *Inconstência 3 - Bloco duplicado*
  - O mesmo bloco aparece em mais de um arquivo.

BL	**1000100100001111**
BU	**0111021011110000**

  - Neste caso, um bloco livre é alocado, o conteúdo do bloco inconsistente é copiado para este bloco e o bloco é mantido em um dos dois arquivos
  - Desta maneira, o sistema de arquivos fica consistente mas provavelmente houve corrupção da informação. O erro deve ser apresentado na console.

Consistência do SA
Verificação em Blocos

- *Inconstência 4 - Bloco usado e livre*
  - O mesmo bloco aparece tanto na lista de blocos em uso como na lista de blocos livres.

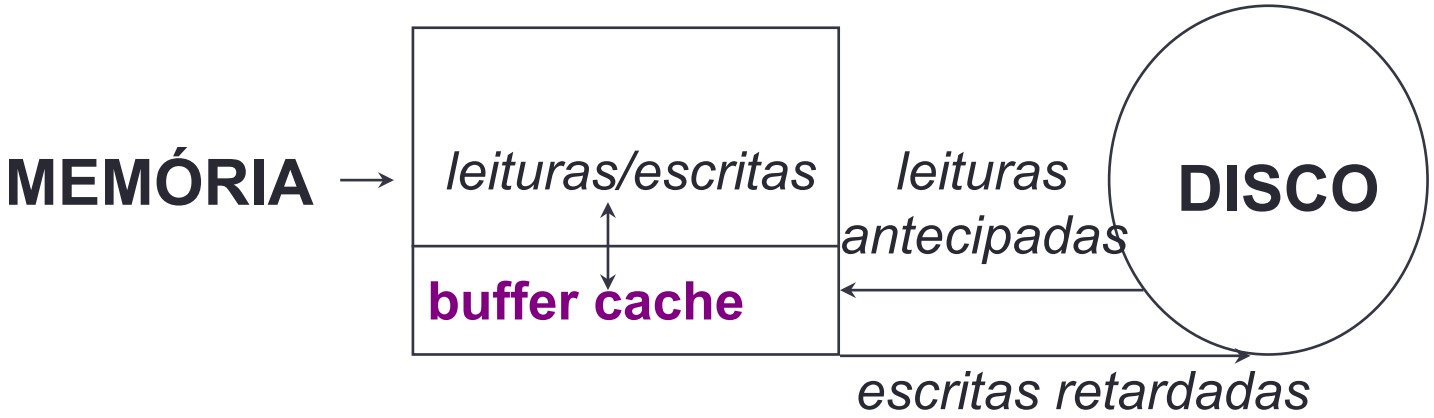


  - Neste caso, remove-se o bloco da lista de blocos livres

Desempenho do Sistema de Arquivos

- Em geral, um acesso a disco é muito mais lento do que um acesso à memória principal.
- Uma maneira simples de aumentar a performance do sistema de arquivos consiste em escrever e ler dados da memória principal sempre que possível.
- Para tanto, necessitamos manter uma área em memória que armazenará temporariamente os blocos de disco mais utilizados. Como essa filosofia se assemelha à filosofia das caches físicas, essa área em memória chama-se cache de disco ou buffer cache

Desempenho do Sistema de Arquivos
Buffer Cache



- A gerência da buffer cache necessita prever o que fazer quando a área reservada para a cache estiver cheia. Da mesma maneira que em um sistema paginado, devemos escolher um bloco para ser retirado da buffer cache.

Desempenho do SA
Buffer cache

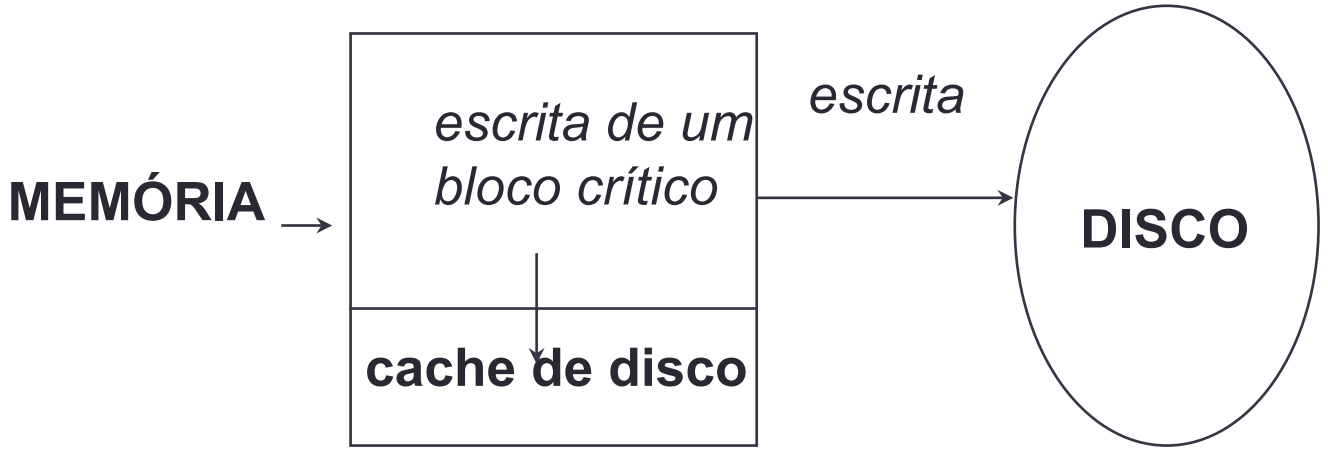
- Este problema, apesar de parecido com o problema de substituição de páginas, não deve ser resolvido da mesma maneira.
- A introdução da buffer cache foi feita para aumentar a performance do sistema de arquivos. Porém, essa solução introduz um complicante na área da confiabilidade do sistema.

Desempenho do SA
Buffer Cache

- Geralmente, uma operação de escrita em arquivo que retorna sucesso indica que o dado foi escrito no disco com sucesso. Em um sistema com buffer cache, isso não é verdade. O retorno de uma operação de escrita simplesmente indica que o dado foi escrito na buffer cache com sucesso. Em tal sistema, dados assumidos pelo usuário como escritos em disco podem ser perdidos após uma pane, porque ainda estavam em memória (na buffer cache).
- Este problema se agrava quando os blocos alterados contém estruturas de controle do sistema de arquivos.
- Assim, geralmente os blocos são divididos em blocos críticos e blocos de dados.

Desempenho do SA
Buffer Cache

- Os blocos críticos são sempre escritos na buffer cache e no disco. Note que há um bypass da cache.



Desempenho do SA
Buffer Cache

- Mesmo para os blocos de dados, não é boa política deixar os blocos alterados por muito tempo em memória. Por esta razão, é geralmente utilizado um algoritmo LRU para substituição mas, além disso, um daemon percorre periodicamente a cache e escreve os blocos com data de modificação superior a *x* segundos no disco.
- No Unix, existe um comando de usuário chamado *sync* que força a escrita dos blocos modificados em cache no disco.