ACH2024

Aula 10
Organização de arquivos, e acesso em memória secundária

Prof Helton Hideraldo Bíscaro

Arquivos

- O que é um arquivo?

Arquivos

- O que é um arquivo?
  - Unidade lógica de informação produzida por processos
  - Pode ser visto como um espaço de endereçamento (em disco)

Arquivos

- O que é um arquivo?
  - Unidade lógica de informação produzida por processos
  - Pode ser visto como um espaço de endereçamento (em disco)
- Quais os tipos de arquivos?

Arquivos

- O que é um arquivo?
  - Unidade lógica de informação produzida por processos
  - Pode ser visto como um espaço de endereçamento (em disco)
- Quais os tipos de arquivos?
  - Texto (.txt, .doc, …)
  - Imagem (.tiff, .jpeg, .gif,…)
  - Planilha (.xls, .csv, .txt)

Arquivos

- Quais são algumas características desejáveis de arquivos (em geral)?

Arquivos

- Quais são algumas características desejáveis de arquivos (em geral)? Algumas das principais:
  - Persistência: conteúdo precisa ser mantido mesmo quando o computador é desligado, isto é, preciso ser armazenado em um dispositivo de memória não volátil (memória secundária), normalmente o disco
  - Concorrência: múltiplos processos devem poder acessar o mesmo arquivo simultaneamente
  - Capacidade de armazenamento: quero armazenar uma grande quantidade de informação, maior do que a capacidade da memória principal (às vezes até mesmo para um ÚNICO arquivo!)

Arquivos

- Quais são algumas características desejáveis de arquivos (em geral)? Algumas das principais:
  - Persistência: conteúdo precisa ser mantido mesmo quando o computador é desligado, isto é, preciso ser armazenado em um dispositivo de memória não volátil (memória secundária), normalmente o disco
  - Concorrência: múltiplos processos devem poder acessar o mesmo arquivo simultaneamente
  - Capacidade de armazenamento: quero armazenar uma grande quantidade de informação, maior do que a capacidade da memória principal (às vezes até mesmo para um ÚNICO arquivo!)
    - Quais tipos de arquivos costumam ser grandes assim?

Arquivos

- Quais são algumas características desejáveis de arquivos (em geral)? Algumas das principais:
  - Persistência: conteúdo precisa ser mantido mesmo quando o computador é desligado, isto é, preciso ser armazenado em um dispositivo de memória não volátil (memória secundária), normalmente o disco
  - Concorrência: múltiplos processos devem poder acessar o mesmo arquivo simultaneamente
  - Capacidade de armazenamento: quero armazenar uma grande quantidade de informação, maior do que a capacidade da memória principal (às vezes até mesmo para um ÚNICO arquivo!)
    - Quais tipos de arquivos costumam ser grandes assim? Arquivos de DADOS, como planilhas (conjunto de registros – linhas, cada registro com seus campos).

Arquivos

- Nas próximas aulas veremos como esses arquivos (conjunto de registros) são organizados no disco, pensando na viabilização de:
  - Leitura sequencial e aleatória
  - Alteração de registros
  - Inserção de registros
  - Exclusão de registros

  Tudo isso lembrando que o arquivo tem que passar pela memória principal (ou seja, sua organização em disco tem que ser compatível com a estrutura da memória principal).
- Ligação direta com as disciplinas de Sistemas Operacionais e Banco de Dados

Estrutura interna de arquivos de dados

- Muitos arquivos de dados são uma lista de dados, cada um podendo conter vários campos
  - Ex: linhas de uma tabela, seja de uma planilha, de um .csv, de uma tabela de banco de dados, ...
  - Cada dado normalmente tem uma chave, pela qual é possível realizar operações de busca e ordenação
- Chamaremos cada dado (contendo a chave e demais campos) de **registro**
- Como separar os campos? Como separar os registros?

Métodos para organização em campos

- Comprimento fixo
- Indicador de comprimento
- Delimitadores
- Uso de *tags* (etiquetas)

Campos com tamanho fixo

- Cada campo ocupa no arquivo um tamanho fixo, pré-determinado
- O fato do tamanho ser conhecido garante que é possível recuperar cada campo
  - Como?

Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

Campos com tamanho fixo

```
struct {
        char last[10];
        char first[10];
        char city[15];
        char state[2];
        char zip[9];
} set_of_fields;
```

Campos com tamanho fixo

- Quais as desvantagens desta abordagem?

Campos com tamanho fixo

- O espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo (desperdício)
  - Solução inapropriada quando se tem uma grande quantidade de dados com tamanho variável
  - Razoável apenas se o comprimento dos campos é realmente fixo ou apresenta pouca variação

Campos com indicador de comprimento

- O tamanho de cada campo é armazenado imediatamente antes do dado

- Desvantagens desta abordagem?

```
05Maria05Rua 103123 10São Carlos
04João05Rua A0325509Rio Claro
05Pedro06Rua 10025610Rib. Preto
```

Campos com indicador de comprimento

- O tamanho de cada campo é armazenado imediatamente antes do dado
  - Se o tamanho do campo é inferior a 256 bytes, o espaço necessário para armazenar a informação de comprimento é um único byte
- Desvantagens desta abordagem?

```
05Maria05Rua 103123 10São Carlos
04João05Rua A0325509Rio Claro
05Pedro06Rua 10025610Rib. Preto
```

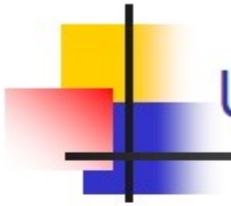
Gasto de espaço para cada campo (vale a pena ou não dependendo da variabilidade do tamanho dos campos

Complica o cálculo da posição de um dado campo

Campos separados por delimitadores

- Caractere(s) especial(ais) (que não fazem parte do dado) são escolhido(s) para ser(em) inserido(s) ao final de cada campo
  - Ex.: para o campo *nome* pode-se utilizar /, tab, #, etc...
  - Espaços em branco não servem na maioria dos casos

```
Maria|Rua 1|123|São Carlos|
João|Rua A|255|Rio Claro|
Pedro|Rua 10|56|Rib. Preto|
```



Uso de uma *tag* do tipo "keyword=value"

- Vantagem: o campo fornece informação semântica sobre si próprio
  - Fica mais fácil identificar o conteúdo do arquivo

- Desvantagem: as *keywords* podem ocupar uma porção significativa do arquivo

```
Nome=Maria|Endereço=Rua 1|Número=123|Cidade=São Carlos|
Nome=João|Endereço=Rua A|Número=255|Cidade=Rio Claro|
Nome=Pedro|Endereço=Rua 10|Número=56|Cidade=Rib. Preto|
```

Métodos para organização em registros

- Tamanho fixo
- Número fixo de campos
- Indicador de tamanho
- Uso de índice
- Utilizar delimitadores

Registros de tamanho fixo

- Analogamente ao conceito de campos de tamanho fixo, assume que todos os registros têm o mesmo tamanho, com campos de tamanho fixo ou não
  - Um dos métodos mais comuns de organização de arquivos

Registro de tamanho fixo e campos de tamanho fixo:

Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

Registro de tamanho fixo e campos de tamanho variável:

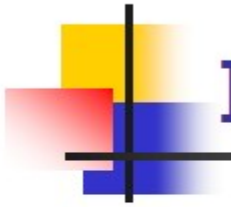
Maria|Rua 1|123|São Carlos| ← Espaço vazio →|
João|Rua A|255|Rio Claro| ← Espaço vazio →|
Pedro|Rua 10|56|Rib. Preto| ← Espaço vazio →|

Registros com número fixo de campos

- Ao invés de especificar que cada registro contém um tamanho fixo, podemos especificar um número fixo de campos
  - O tamanho do registro é variável
  - Neste caso, os campos seriam separados por delimitadores

Registro com número fixo de campos:

Maria|Rua 1|123|São Carlos|João|Rua A|255|Rio Claro|Pedro|Rua 10|56|Rib. Preto|



Indicador de tamanho para registros

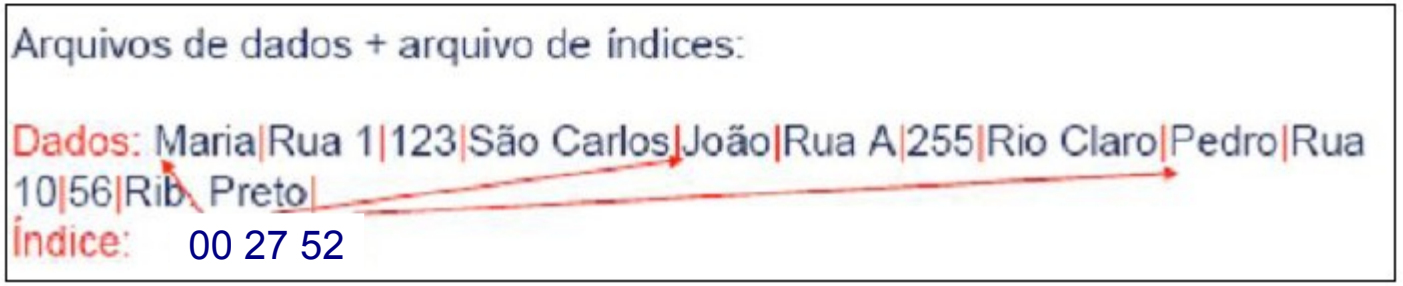
- O indicador que precede o registro fornece o seu tamanho total
  - Os campos são separados internamente por delimitadores
  - Boa solução para registros de tamanho variável

Registro iniciados por indicador de tamanho:

27Maria|Rua 1|123|São Carlos|25João|Rua A|255|Rio Claro|27Pedro|Rua 10|56|Rib. Preto|

Utilizar um índice

- Um índice externo poderia indicar o deslocamento de cada registro relativo ao início do arquivo
  - Pode ser utilizado também para calcular o tamanho dos registros
  - Os campos seriam separados por delimitadores



Utilizar delimitadores

- Separar os registros com delimitadores análogos aos de fim de campo
  - O delimitador de campos é mantido, sendo que o método combina os dois delimitadores
  - Note que delimitar fim de campo é diferente de delimitar fim de registro

Registro delimitado por marcador (#):

Maria|Rua 1|123|São Carlos|#João|Rua A|255|Rio Claro|#Pedro|Rua 10|56|Rib. Preto|

Cabeçalho de arquivos

- Cabeçalho do arquivo (descritor) pode conter informações como:
  - Descrição dos formatos dos campos de um registro
  - Códigos de tipos de registros para registros de tamanho variável
  - Data de criação e atualização

Permite definição de padrões (ex: pdf, tiff, jpg) e facilita conversão entre padrões

Armazenamento não volátil de arquivos

- Quando estamos falando em arquivo, normalmente estamos falando em armazenamento em memória SECUNDÁRIA
- Memória secundária:
  - HD
  - Disquetes
  - CD-ROM
  - DVD
  - Pen-drives
  - Chips de memória
  - Fita magnética
  - ...

Memória primária x secundária

Mem. volátil

Mem. secundária

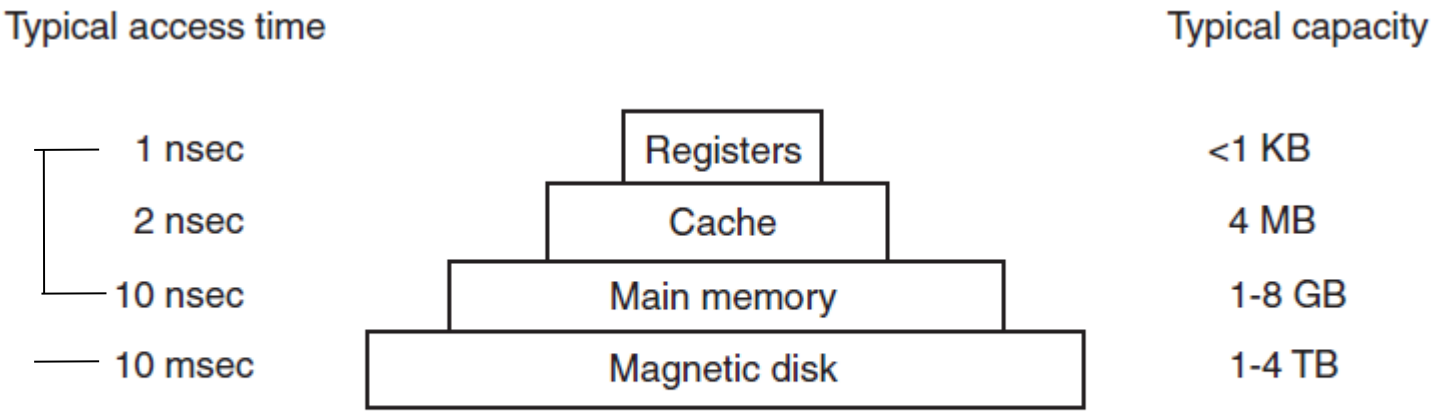


Figure 1-9. A typical memory hierarchy. The numbers are very rough approximations.

(TANEMBAUM & BOS, 2015)

Discos X Memória Principal

- Capacidade de Armazenamento
  - HD – muito alta, a um custo relativamente baixo
  - RAM – limitada pelo custo e espaço
- Tipo de Armazenamento
  - HD – não volátil
  - RAM – volátil

Discos X Memória Principal

- Capacidade de Armazenamento
  - HD – muito alta, a um custo relativamente baixo
  - RAM – limitada pelo custo e espaço
- Tipo de Armazenamento
  - HD – não volátil
  - RAM – volátil

ENTÃO POR QUE USAR MEMÓRIA PRIMÁRIA???

Discos X Memória Principal

- Tempo de acesso
  - HD: ~ microsegundos μs (10^{-6})
  - RAM: ~ nanosegundos ηs (10^{-9})
  - HDs são centenas – e até milhares – de vezes mais lentos que memória RAM

Discos X Memória Principal

- Tempo de acesso
  - HD: ~ microsegundos μs (10^{-6})
  - RAM: ~ nanosegundos ηs (10^{-9})
  - HDs são centenas – e até milhares – de vezes mais lentos que memória RAM

POR QUE ESSA DIFERENÇA?

VAMOS ENTENDER COMO SE DÁ A LEITURA DE UM DADO NO HD...

HD – Hard Disk (em contraposição às demais mídias da época)

HD – Hard Disk (em contraposição às demais mídias da época)

- No início, mais em sistemas corporativos
  - 1.70m de altura e de comprimento, quase 1 tonelada
  - Chamado "unidade de disco"

IBM 350 (1956)

HDs

- HD
  - Em 1973, IBM lançou o que é considerado o pai dos HDs modernos
    - Winchester

Estrutura de um HD

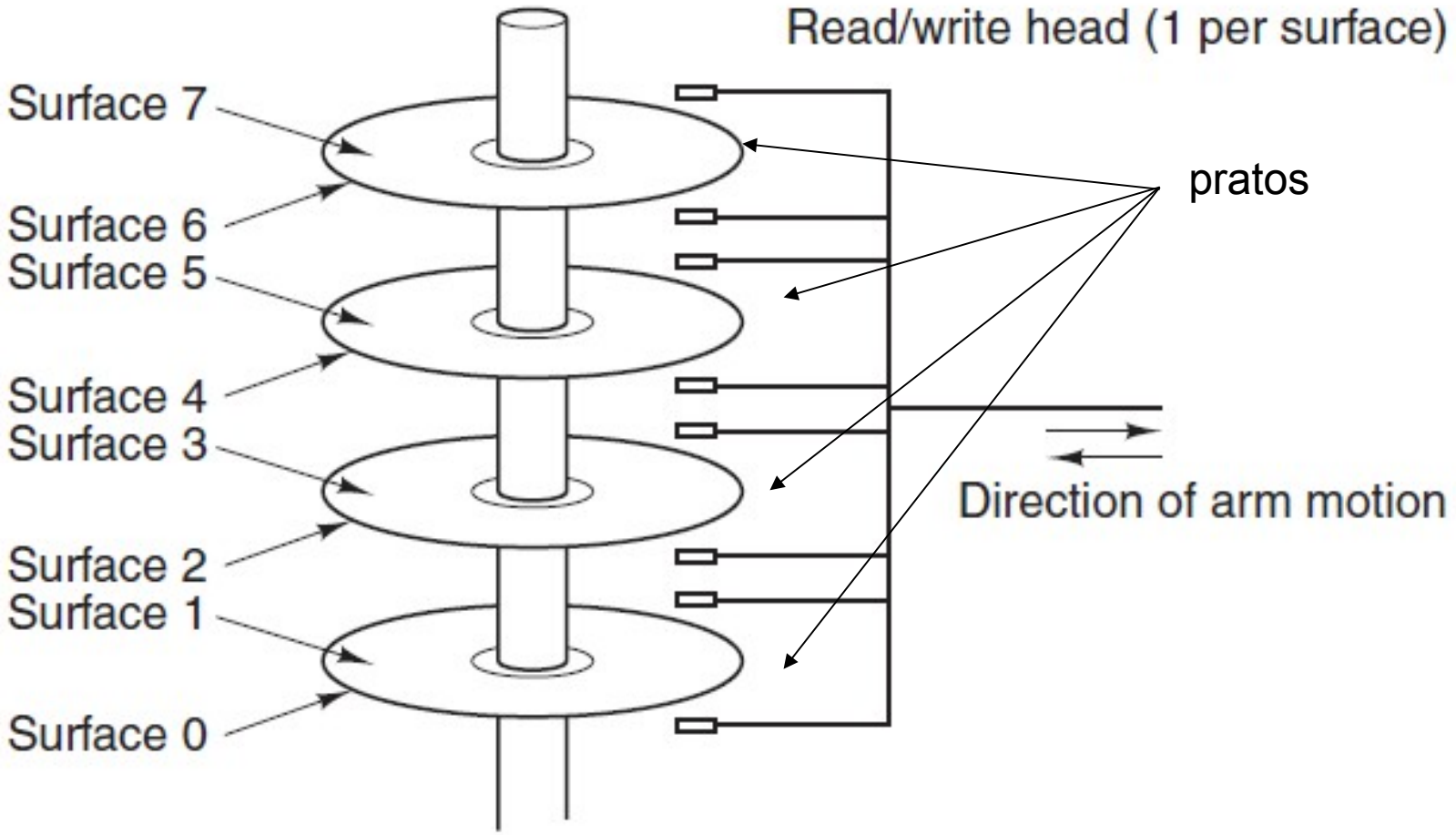
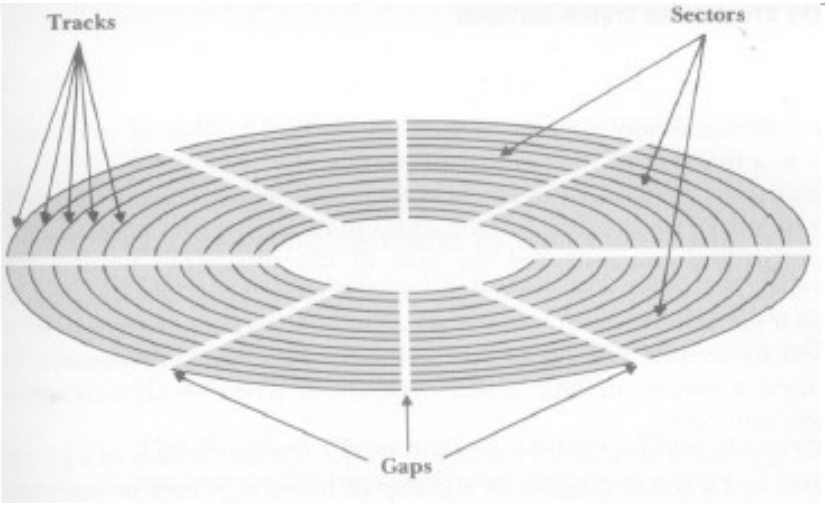


Figure 1-10. Structure of a disk drive.

(TANEMBAUM & BOS, 2015)

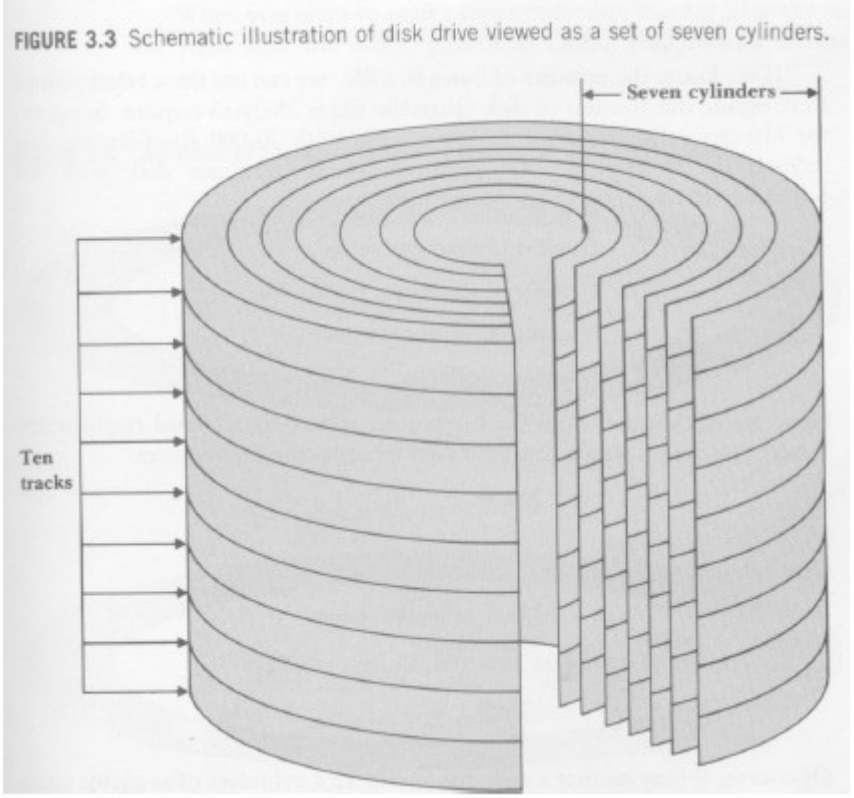


Organização da informação no disco

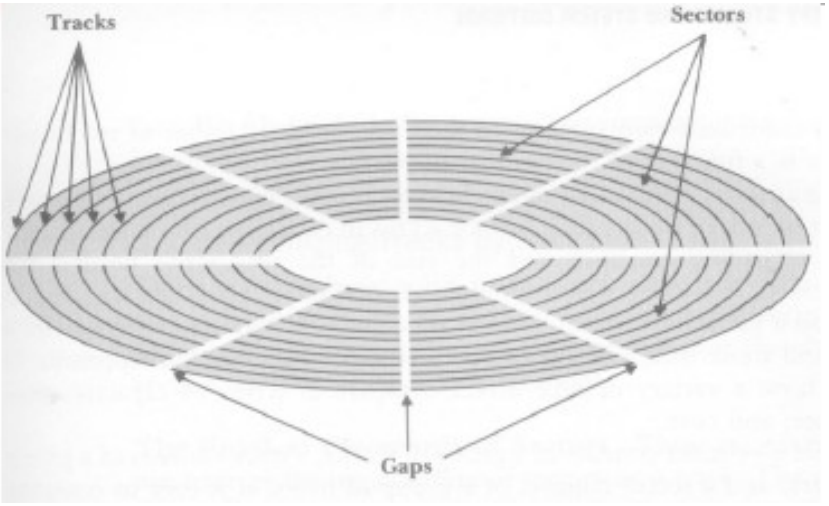
- Disco: conjunto de 'pratos' empilhados
  - Dados são gravados nas superfícies desses pratos
- Superfícies: são organizadas em trilhas
- Trilhas: são organizadas em setores
- Cilindro: conjunto de trilhas na mesma posição

Um **setor** é a menor porção endereçável do disco

A divisão de uma trilha em setores é definida pelo disco, e não pode ser mudada.



O você acha que o computador deve fazer para ler algum dado do HD?



O você acha que o computador deve fazer para ler algum dado do HD?

1) Identifica em que setor/trilha/superfície está a informação

2) Movimenta o braço de leitura para o cilindro correto (para poder acessa a trilha correta)

3) Rotaciona o prato para posicionar a cabeça de leitura sobre o setor correto

4) Faz a leitura de um certo nr de bytes

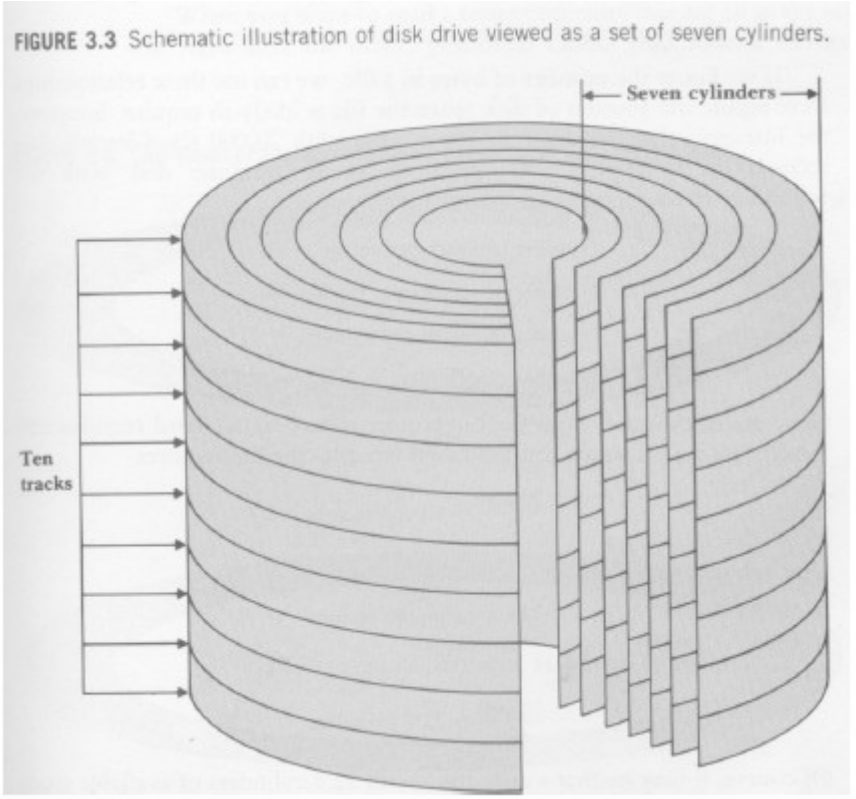
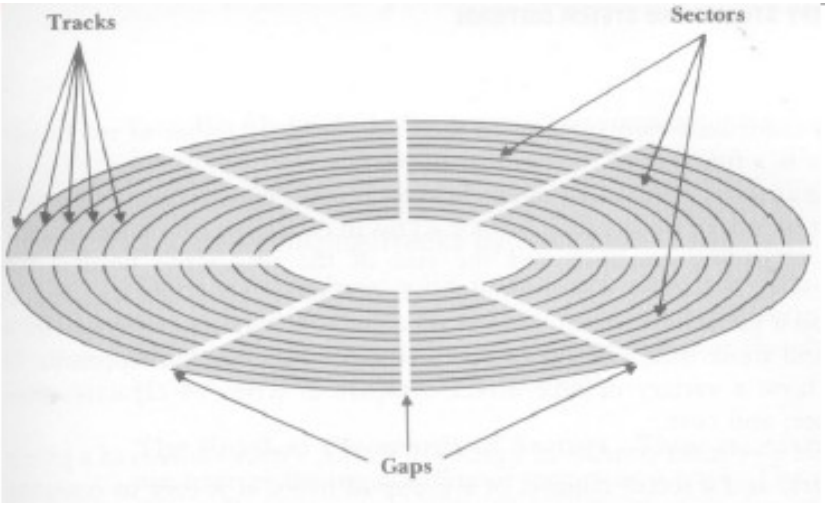


FIGURE 3.3 Schematic illustration of disk drive viewed as a set of seven cylinders.



O você acha que o computador deve fazer para ler algum dado do HD?

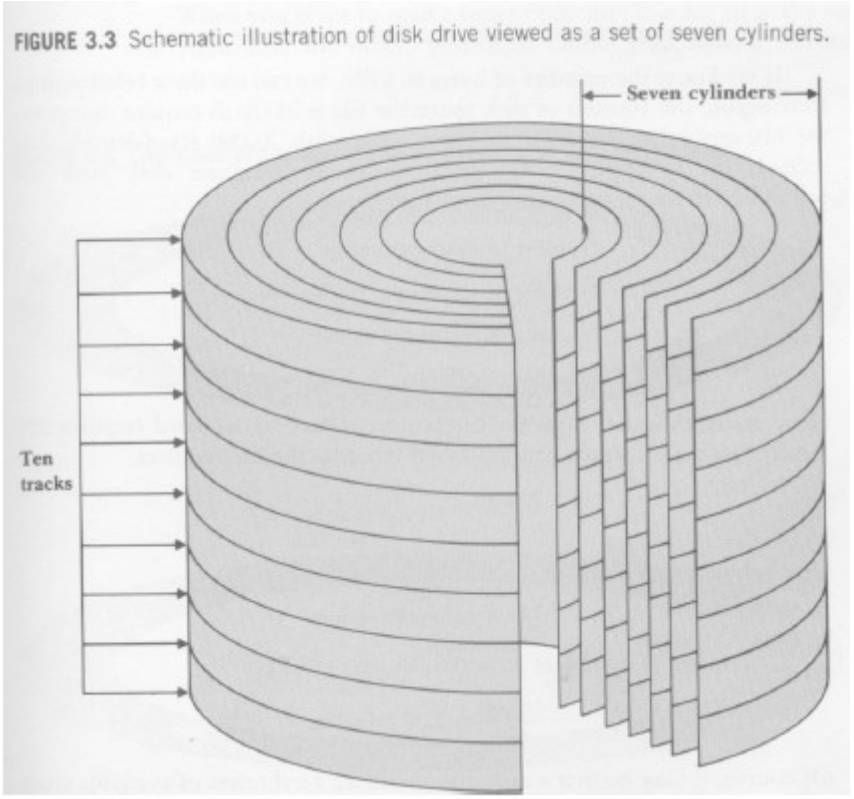
1) Identifica em que setor/trilha/superfície está a informação

2) Movimenta o braço de leitura para o cilindro correto (para poder acessa a trilha correta)

3) Rotaciona o prato para posicionar a cabeça de leitura sobre o setor correto

4) Faz a leitura de um certo nr de bytes

Passos 2 e 3 (chamados SEEK) são mecânicos! Por isso demora tanto!



Seeking

- Movimento de posicionar a cabeça de L/E sobre a trilha/setor desejado
- O conteúdo de todo um cilindro pode ser lido com 1 único *seeking*
- É o movimento mais lento da operação leitura/escrita
- Deve ser reduzido ao mínimo

Observação: Acesso sequencial x acesso aleatório (ou randômico ou direto) – com relação ao DISPOSITIVO

- Apesar do custo de um seek, o acesso é direto, pois não é necessário ler dados anteriores
  - Também chamado de acesso aleatório ou randômico
- Em contraposição, fitas demandam acesso sequencial, ou seja, é preciso passar por todo o trecho de fita anterior ao que contém o dado desejado

Quanto mais seeks, mais cara a leitura

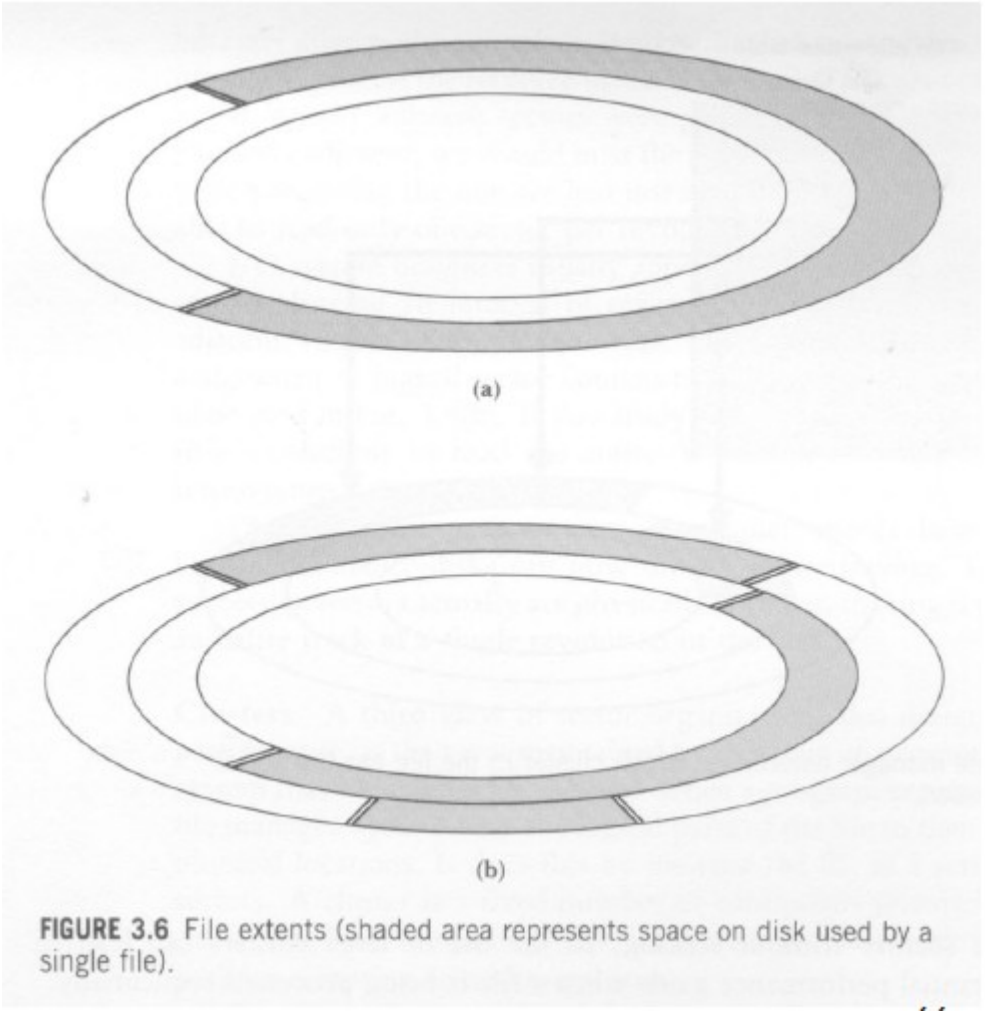


FIGURE 3.6 File extents (shaded area represents space on disk used by a single file).

Os arquivos não são estáticos, por isso podem não estar armazenados de forma contígua pelo disco

→ tem que armazenar pedaços dos arquivos nos setores (mais ou menos como se fosse uma lista ligada)

O tempo de acesso a uma informação, na prática, depende:

- da distribuição dos dados de um arquivo pelo disco
- da tecnologia

Como calcular tempo de acesso

- Se acesso a disco é caro e queremos escolher estruturas de dados que diminuam o tempo de acesso, precisamos poder calcular (ou estimar) o tempo de acesso de uma forma não muito complicada, pelo menos:

  - independente da distribuição e localização do arquivo em disco

  - independente da tecnologia

- Para simplificar os cálculos, podemos fazer a seguinte aproximação (pior caso):

  - considera-se que é necessário um *seek* por "pedaço" de arquivo a ser lido:

    1) considerando que eu não preciso ler o arquivo inteiro em um dado instante, pois posso estar interessado em apenas um "pedaço"

    2) como há outros processos sendo executados, quando eu quiser ler outro "pedaço" do meu arquivo a cabeça de leitura do disco pode não estar no mesmo lugar onde parou a última leitura desse meu arquivo

  - tempo de acesso total = nr de acessos (*seeks*) * tempo de um acesso

Como calcular tempo de acesso

- Se acesso a disco é caro e queremos escolher estruturas de dados que diminuam o tempo de acesso, precisamos poder calcular (ou estimar) o tempo de acesso de uma forma não muito complicada, pelo menos:

  - independente da distribuição e localização do arquivo em disco

  - independente da tecnologia

- Para simplificar os cálculos, podemos fazer a seguinte aproximação (pior caso):

  - considera-se que é necessário um *seek* por "pedaço" de arquivo a ser lido:

    1) considerando que eu não preciso ler o arquivo inteiro em um dado instante, pois posso estar interessado em apenas um "pedaço"

    2) como há outros processos sendo executados, quando eu quiser ler outro "pedaço" do meu arquivo a cabeça de leitura do disco pode não estar no mesmo lugar onde parou a última leitura desse meu arquivo

  - tempo de acesso total = nr de acessos (*seeks*) * tempo de um acesso

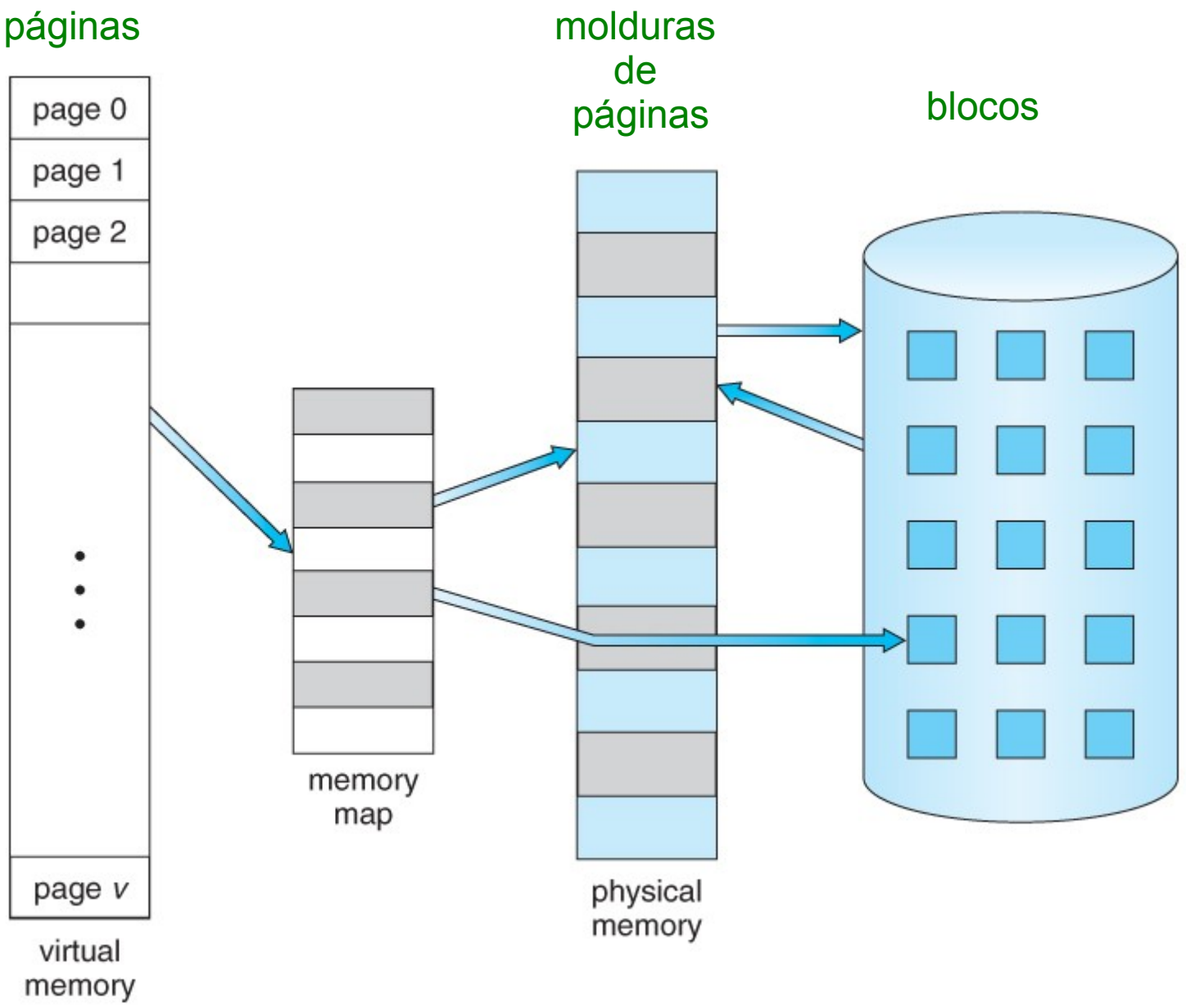
De que tamanho tem que ser esse pedaço?
Pode ser um byte? E se meu programa quiser ler um byte de cada vez?
E quando eu leio um "pedaço", armazeno onde essa informação?

Paginação

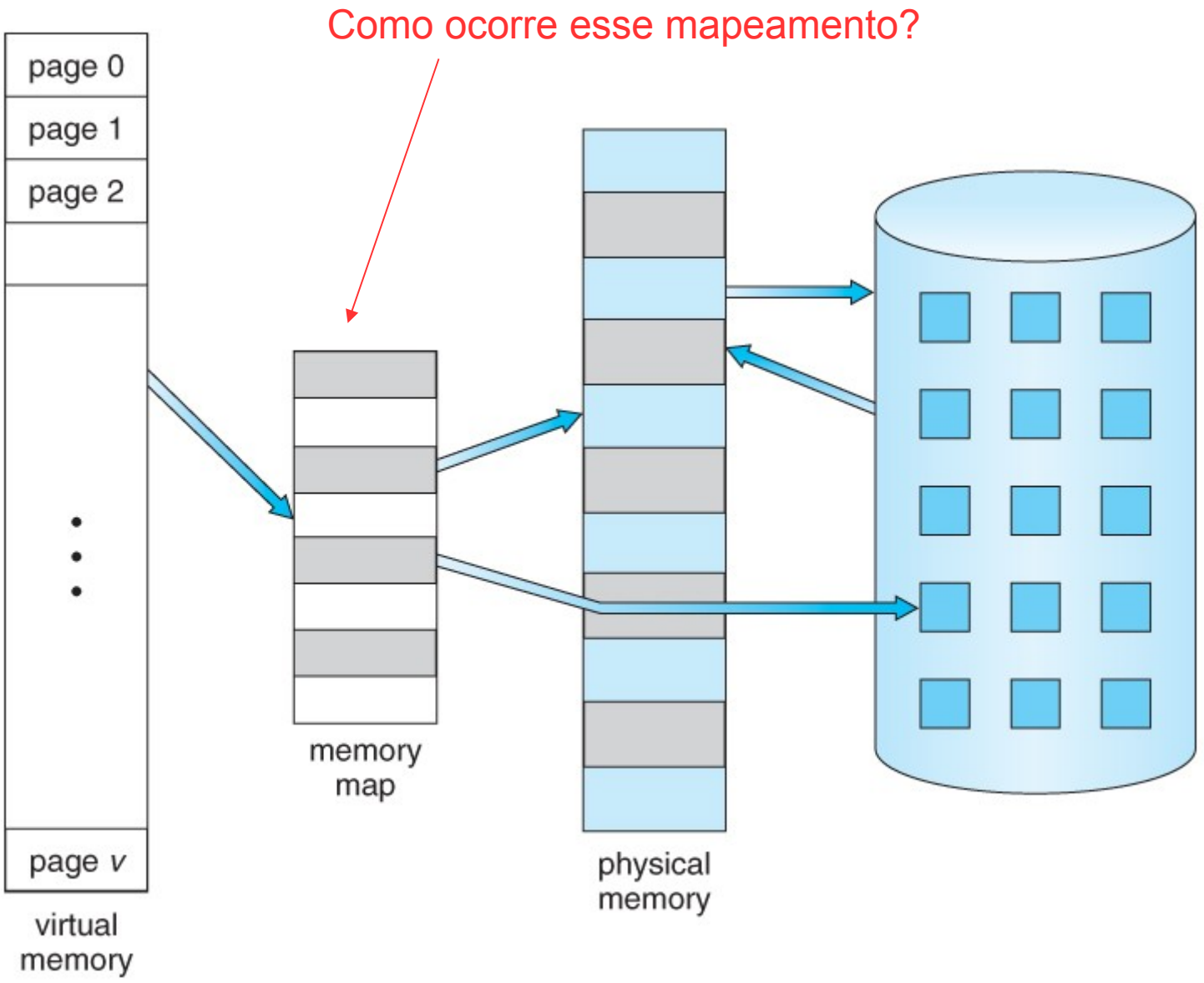
- Disco é grande, mas o acesso é lento
- Fazer várias pequenas leituras no disco tornaria os programas inviáveis…
- Solução: ler um “pedaço” razoável do disco, trazer para a memória principal, processá-la lá conforme necessário, se for salvar reescrever o pedaço todo no disco novamente
- Ou seja, tenho um subconjunto do meu disco em memória principal
- O conteúdo desse “pedaço”, contendo x setores (x um número inteiro), será virtualmente chamado de “página”, e será armazenado fisicamente em um “pedaço” da memória chamado de “moldura de página”
- O tamanho de uma página (que é igual ao tamanho de uma moldura de página) é definida pelo Sistema Operacional (SO) durante a formatação do disco, e não pode ser alterada dinamicamente.

Memória virtual

- O Sistema Operacional (SO) gerencia se a informação (ou seja, a página que contém informação) já está em memória
  - Se não estiver, precisa carregá-la (em uma moldura de página, ie, trecho que memória principal destinado a armazenar uma "página" - ou bloco - do disco)
  - Se não tiver moldura disponível, precisa descarregar alguma no disco antes
- Com isso os programas podem endereçar todo o disco como se ele estivesse em memória principal (memória virtual)



https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html



https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html

Memória virtual – mapeamento de endereços

Seja N o espaço de endereçamento (lógico – tudo o que eu quero endereçar) e M o espaço de memória principal (física)

f : N → M é um mapeamento podendo N >> M

Como fazer esse mapeamento de endereço lógico (do programa) a um endereço físico da memória principal?

Memória virtual – mapeamento de endereços

Lembrando que pode |N| >> |M|

Um endereço de N (lógico, virtual): parte dos bits é o **número** da página (p) e a outra parte é a posição (*offset*) dentro da página (b)

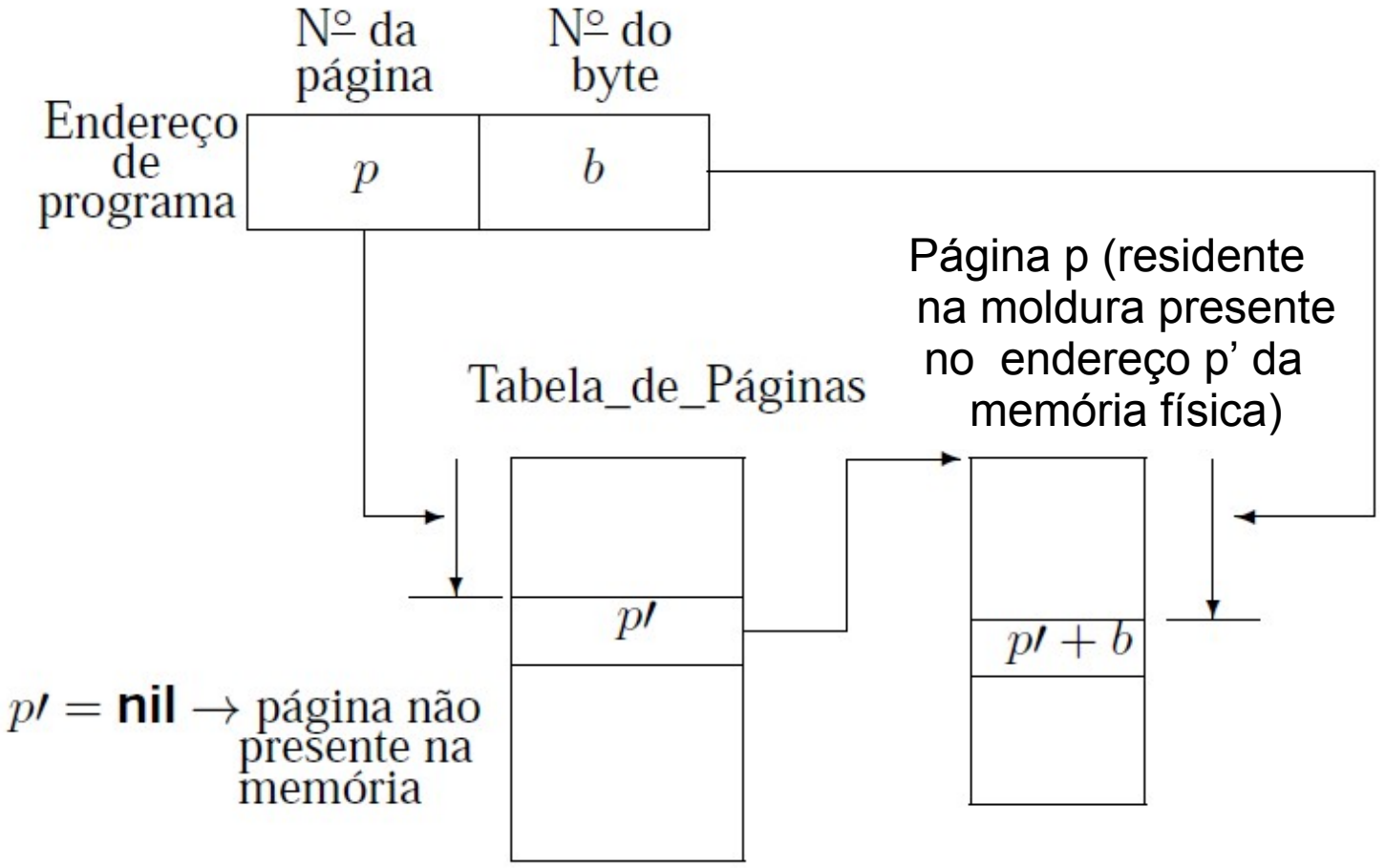
Lembrando que pode haver tantas páginas quanto forem necessárias para cobrir N

Tabela de Páginas: se a página de **número** p estiver na memória principal, a p-ésima entrada da Tabela de Páginas contém o **endereço da moldura** p' (na memória principal) que contém a página p

Isto é, um endereço n pertencente a N é pb (p concatenado com b) e o mapeamento f é:

f(n) = f(pb) = PageTable[p] + b = p' + b

Memória virtual – mapeamento de endereços



Memória Virtual: Reposição de Páginas

- Se não houver uma moldura de página vazia $\rightarrow$ uma página deverá ser removida da memória principal.
- Ideal $\rightarrow$ remover a página que não será referenciada pelo período de tempo mais longo no futuro.
  - tentamos inferir o futuro a partir do comportamento passado.

Memória Virtual: Políticas de Reposição de Páginas

- **Menos Recentemente Utilizada (LRU):**
  - um dos algoritmos mais utilizados,
  - remove a página menos recentemente utilizada,
  - parte do princípio que o comportamento futuro deve seguir o passado recente.
- **Menos Freqüentemente Utilizada (LFU):**
  - remove a página menos feqüentemente utilizada,
  - inconveniente: uma página recentemente trazida da memória secundária tem um baixo número de acessos e pode ser removida.
- **Ordem de Chegada (FIFO):**
  - remove a página que está residente há mais tempo,
  - algoritmo mais simples e barato de manter,
  - desvantagem: ignora o fato de que a página mais antiga pode ser a mais referenciada.

Referências

- Slides da Profa. Graça (ICMC) - http://wiki.icmc.usp.br/index.php/SCC-203_(gracan) (Arquivos 8, 9 e 12)

- Slides do cap 6 do Ziviani

- GOODRICH et al, **Data Structures and Algorithms in C++.** Ed. John Wiley & Sons, Inc. 2nd ed. 2011. Seção 14.2

- ELMARIS, R.; NAVATHE, S. B. **Fundamentals of Database Systems**. 4 ed. Ed. Pearson-Addison Wesley. Cap 13 (até a seção 13.7).

- TANEMBAUM, A. S. & BOS, H. **Modern Operating Systems**. Pearson, 4th ed. 2015