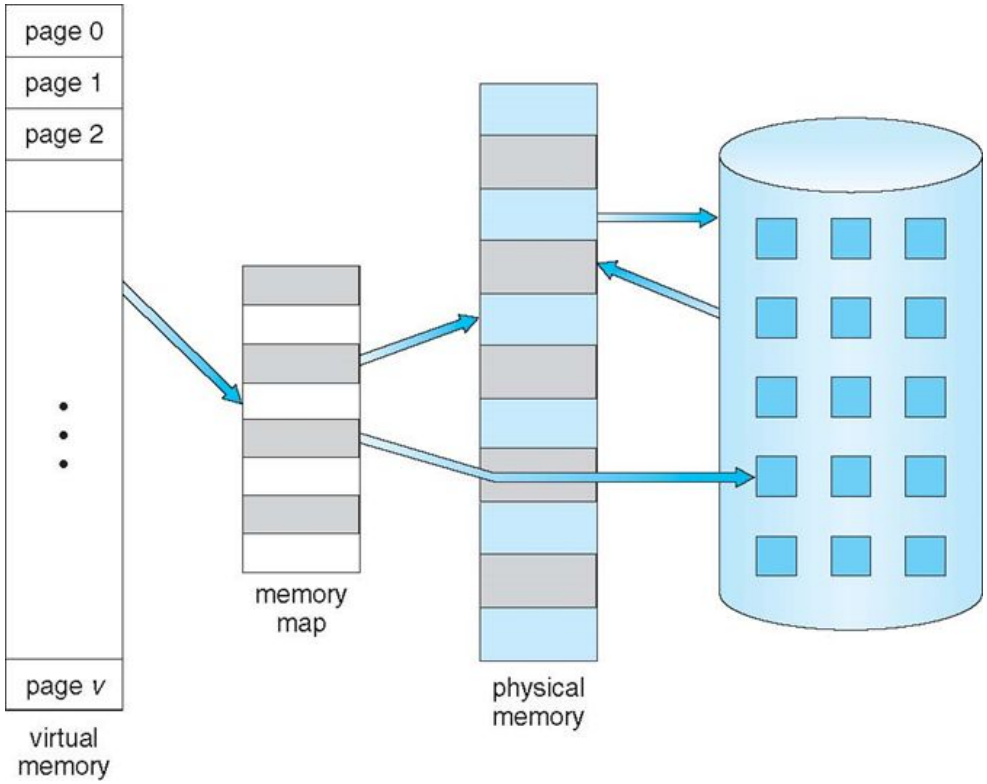
Virtual Memory

André Ferreira - João Victor Bravo
- Tiago Valença

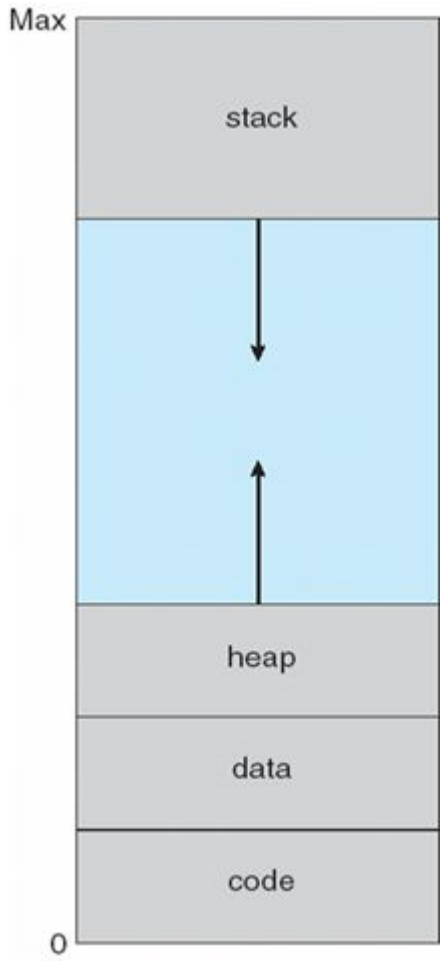
Background

- Memória Virtual – Separação entre a memória lógica do usuário da memória física.
  - Apenas parte do programa precisa estar na memória para ser executado
  - Espaço de endereço lógico pode ser muito maior que o espaço de endereço físico
  - Permite que os endereços de memória sejam divididos por vários processos
  - Permite uma criação de processos mais eficiente
- Como pode ser implementada:
  - Demand paging
  - Demand segmentation

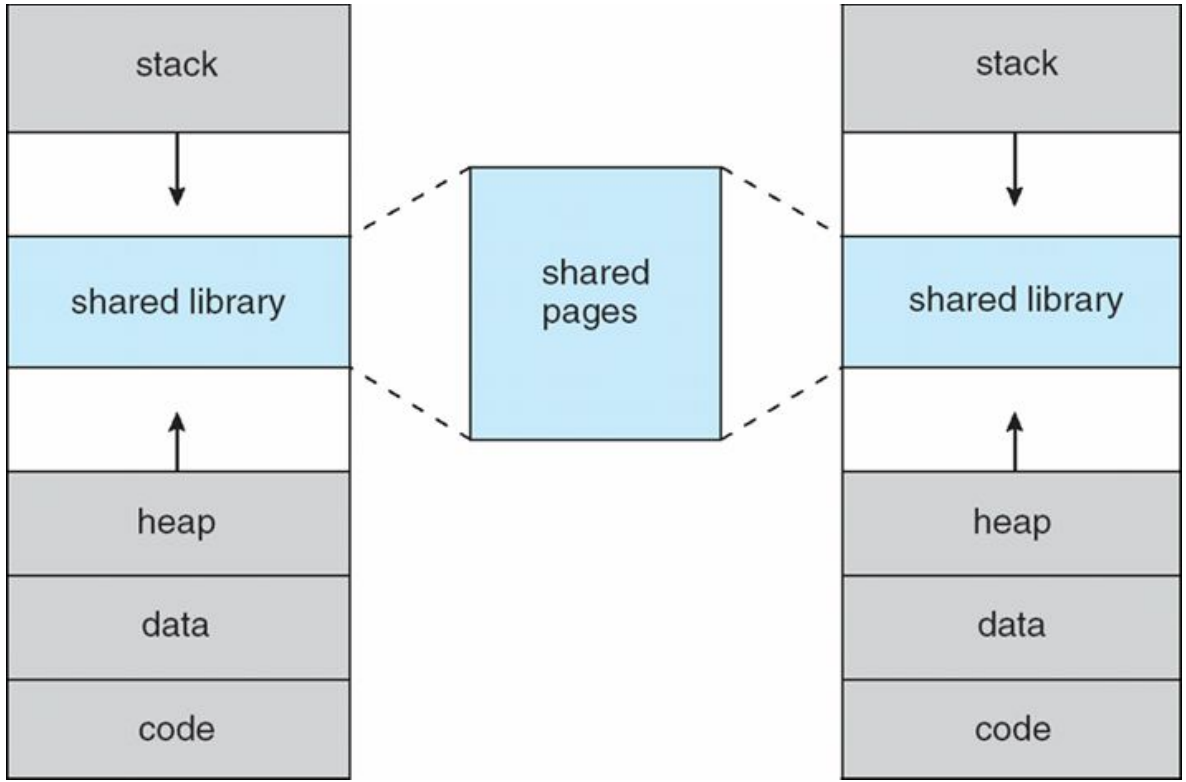
Memória Virtual que é maior que a Memória Física



Endereço de Memória Virtual



Biblioteca dividida utilizando Memória Virtual

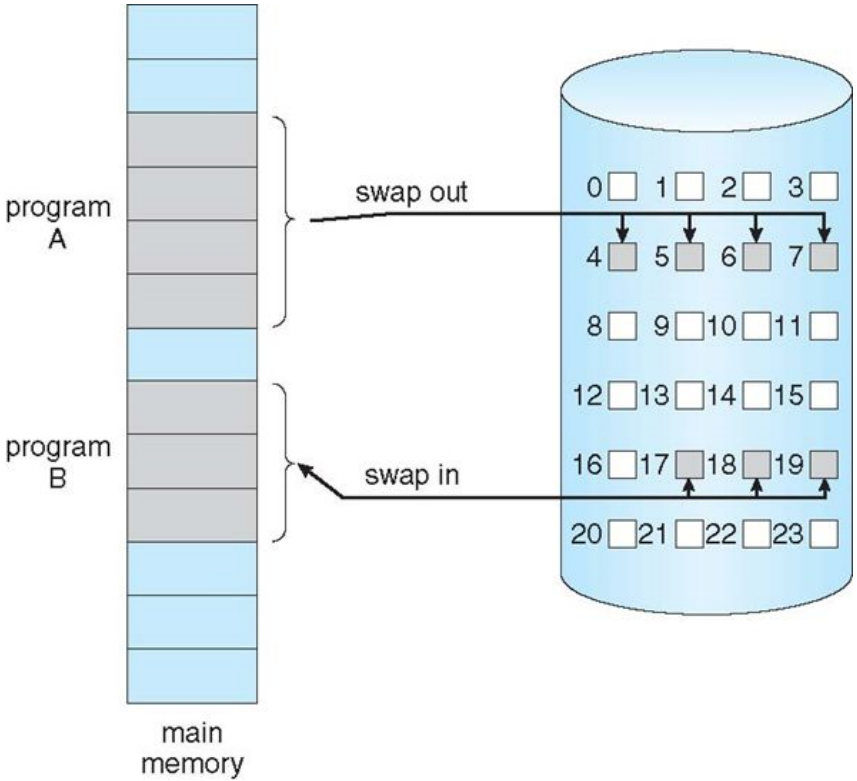


Demand Paging

- Trás uma página para a memória apenas quando necessário
  - Menos E/S necessário
  - Menos memória necessária
  - Resposta mais rápida (não precisa esperar que todas as páginas sejam carregadas)
  - Mais usuários

- Página é necessária ⇨ referência a ela
  - Referência inválida ⇨ abortar
  - Não está na memória ⇨ trazer para a memória
  - **Lazy swapper** – Nunca troca uma página em memória a não ser que seja necessário
  - Swapper que lida com páginas é um **pager**

Transferência de memória paginada para espaço contíguo de disco



Bit Válido-Inválido

- Com cada entrada na tabela de páginas um bit válido–inválido é associado (**v** ⇨ in-memory, **i** ⇨ not-in-memory)
- Inicialmente o bit válido–inválido é colocado como **i** em todas as entradas
- Durante a tradução do endereço, se o bit válido–inválido na entrada da tabela de páginas é **I** ⇨ page fault

Exemplo de um snapshot de uma tabela de páginas:

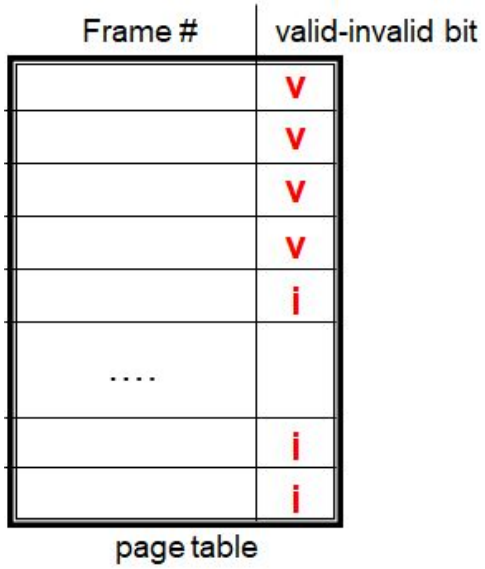
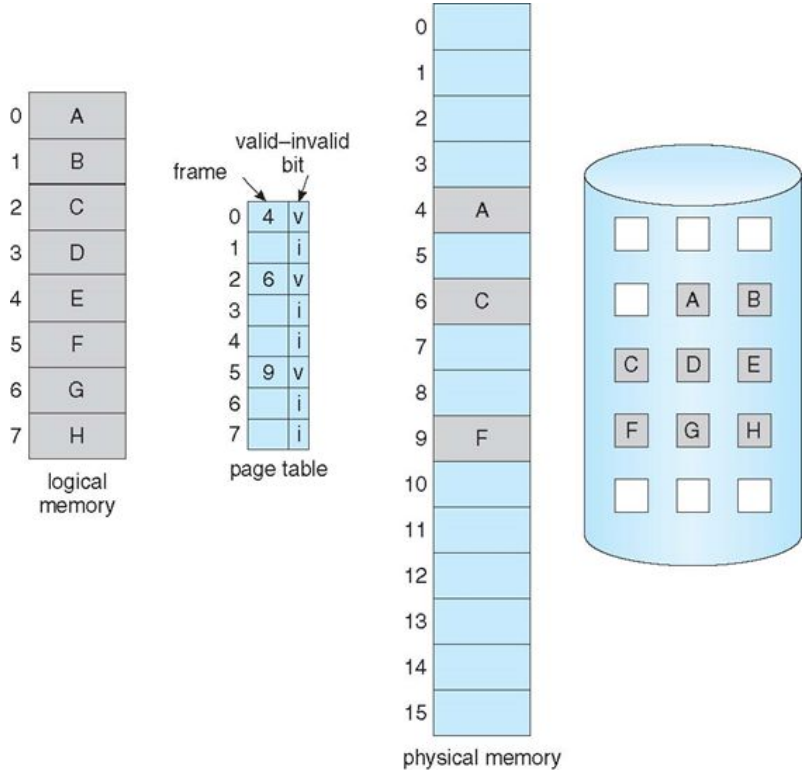


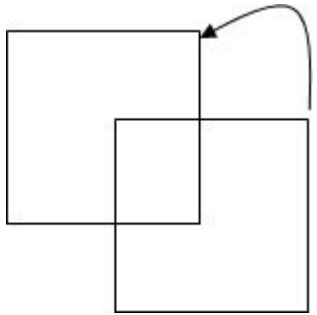
Tabela de Páginas quando algumas páginas não estão na memória principal



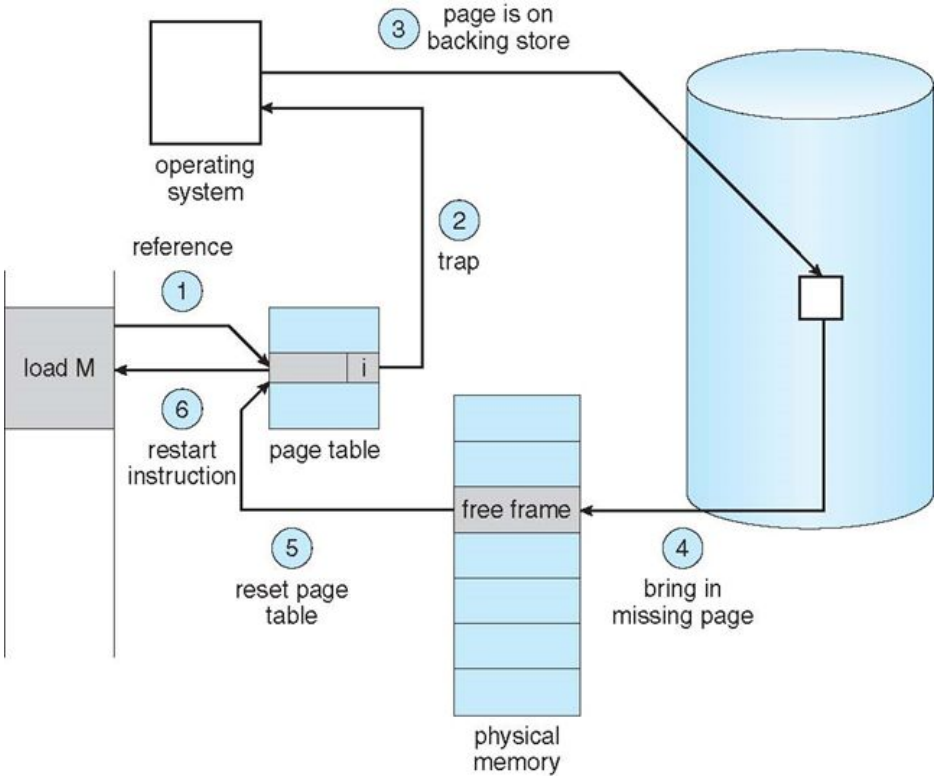
Page Fault

- Se existe uma referência para uma página, a primeira referência aquela página vai travar o sistema operacional: **page fault**

1. Sistema operacional vai olhar em outra tabela para decidir:
   a. Referência Inválida ⇨ abortar
   b. Apenas não está na memória
2. Pegar o frame vazio
3. Trocar a página para o frame
4. Resetar tabelas
5. Colocar o bit de validação para = **v**
6. Reiniciar a instrução que causou o page fault
   a. Bloquear o movimento
   b. Incrementar/decrementar localização automaticamente



Passos para lidar com um Page Fault



Performance de Demand Paging

- Taxa de Page Fault 0 <= *p* <= 1.0
  - se *p* = 0 não tem page faults
  - se *p* = 1, toda referência é uma fault

- Effective Access Time (EAT)

  EAT = (1 – *p*) x memory access

  \+ *p* (page fault overhead

  + swap page out

  + swap page in

  + restart overhead)

Exemplo de Demand Paging

- Tempo de acesso a memória = 200 nanosegundos
- Tempo médio de serviço de um page-fault = 8 milissegundos

  EAT = (1 – p) x 200 + p (8 milliseconds) #EAT é calculado em microsegundos

  EAT = (1 – p) x 200 + p x 8,000,000

  EAT = 200 + p x 7,999,800

- Se um acesso a cada 1,000 causar um page fault, então

  EAT = 8.2 microsegundos

  É uma diminuição de velocidade num fator de 40!!

Para manter a redução de velocidade em 10% p < 0.0000025 (1 em ~400 mil)

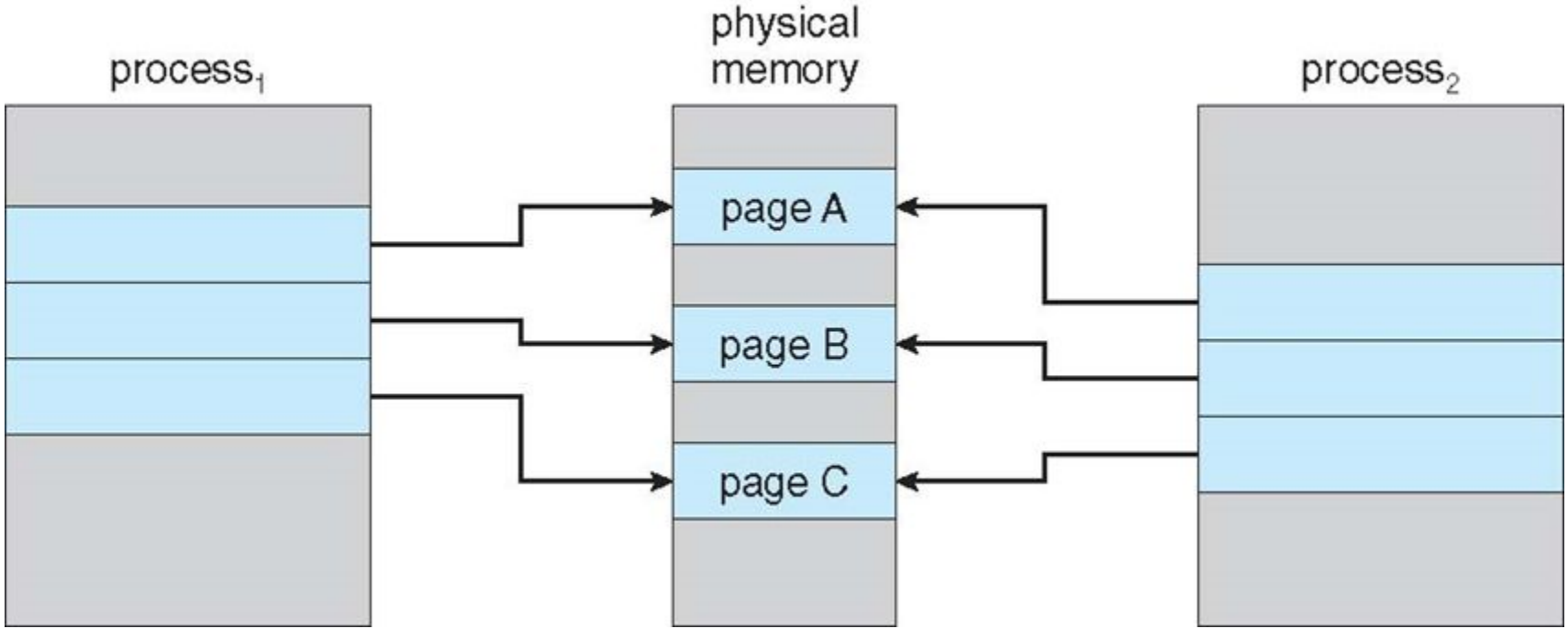
Criação de Processos

- Memória Virtual permite outros benefícios durante a criação de processos:
  - Copy-on-Write
  - Memory-Mapped Files (Veremos depois)

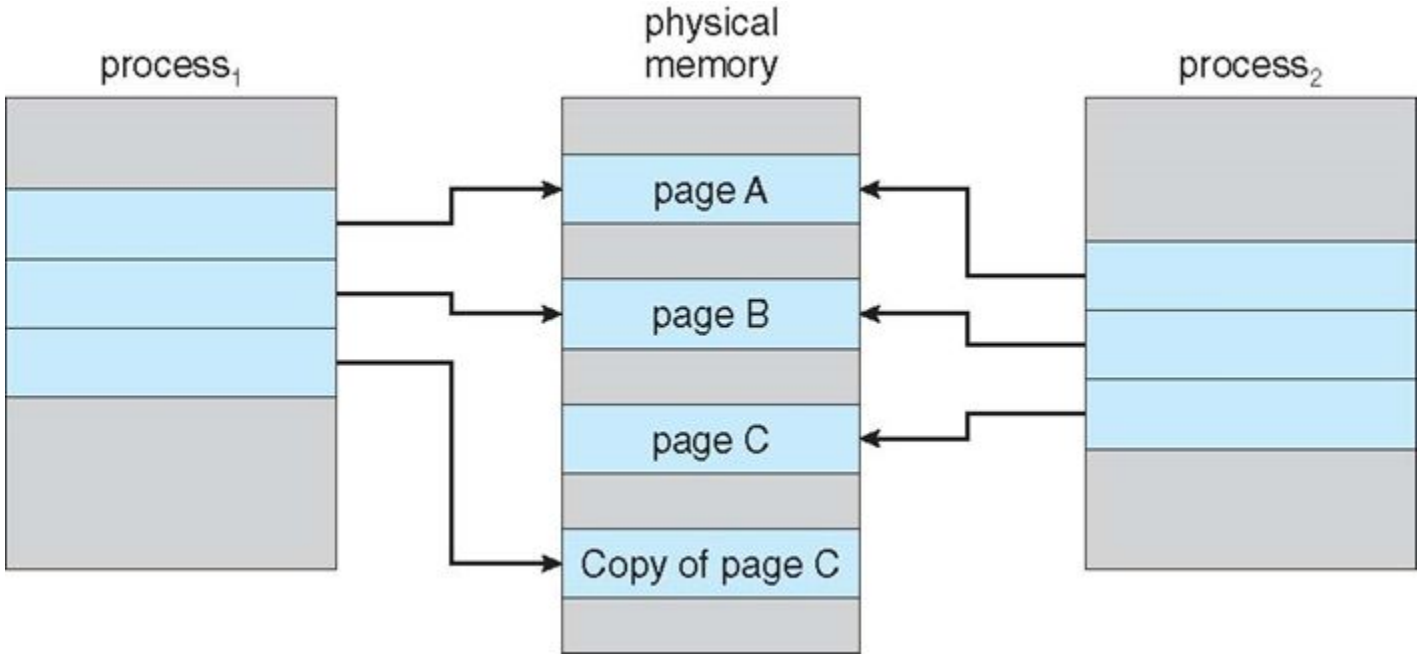
Copy-On-Write

- Copy-on-Write (COW) permite que ambos os processos pais e filhos inicialmente dividam as mesmas páginas na memória
  - Se qualquer um dos processos modificar uma página compartilhada, apenas nesse momento a página é copiada
- COW permite uma criação de processos mais eficiente já que apenas páginas modificadas são copiadas
- Páginas livres são alocadas de uma **pool** de páginas “zeroed-out”

Antes do processo 1 modificar a página C



Depois do processo 1 modificar a página C

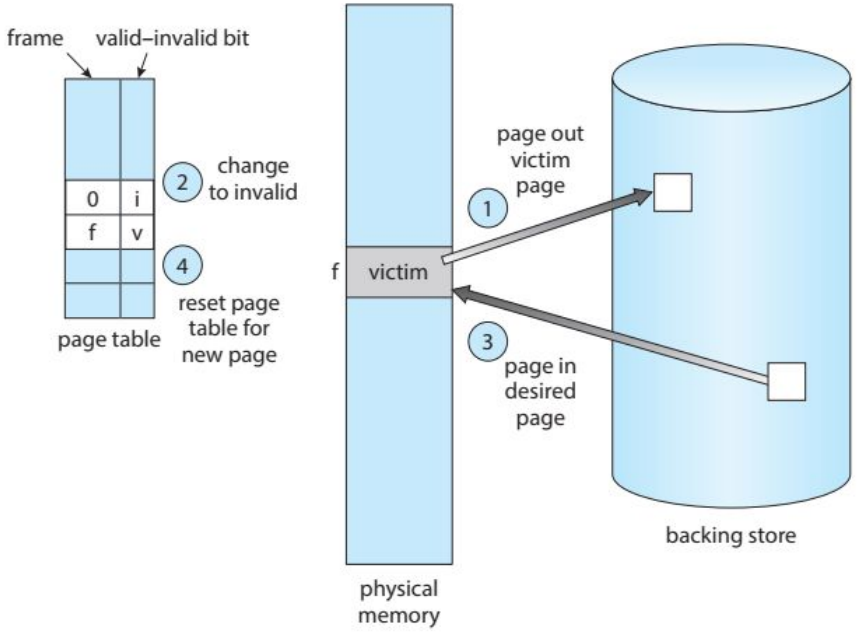


O que acontece se não existir frame livre?

- Page replacement – Encontra uma página na memória, mas que não esteja em uso, troque ela
  - algoritmo
  - desempenho – queremos um algoritmo que vai resultar no menor número possível de page faults
- A mesma página pode ser trazida para a memória múltiplas vezes

Remanejamento de Páginas

- Esquema básico

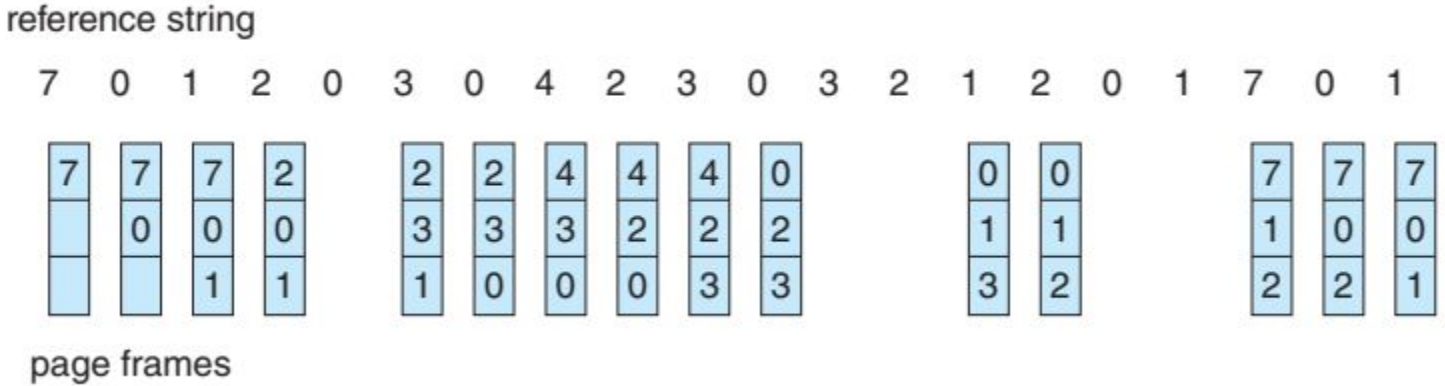


Remanejamento de Páginas

- Algoritmos de Remanejamento
  - *FIFO*;
  - *Optimal Page Replacement*;
  - *Least Recently Used* (LRU);
  - *LRU-Approximation*;
  - *Counting-based*;
  - *Page-buffering*;

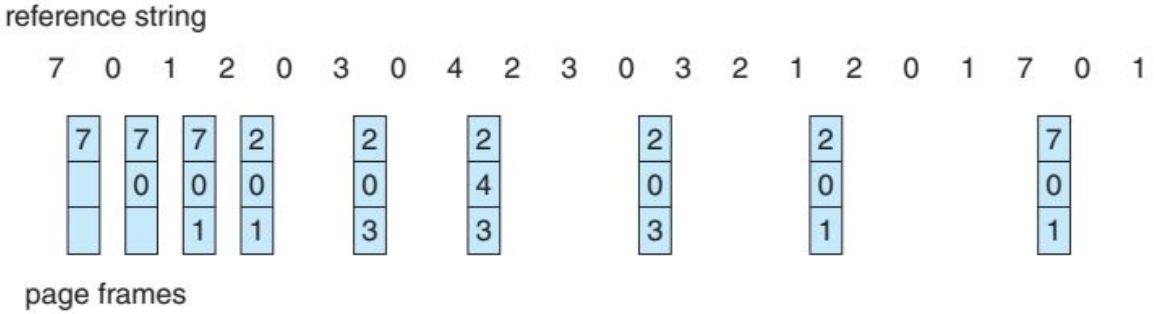
Remanejamento de Páginas

- FIFO



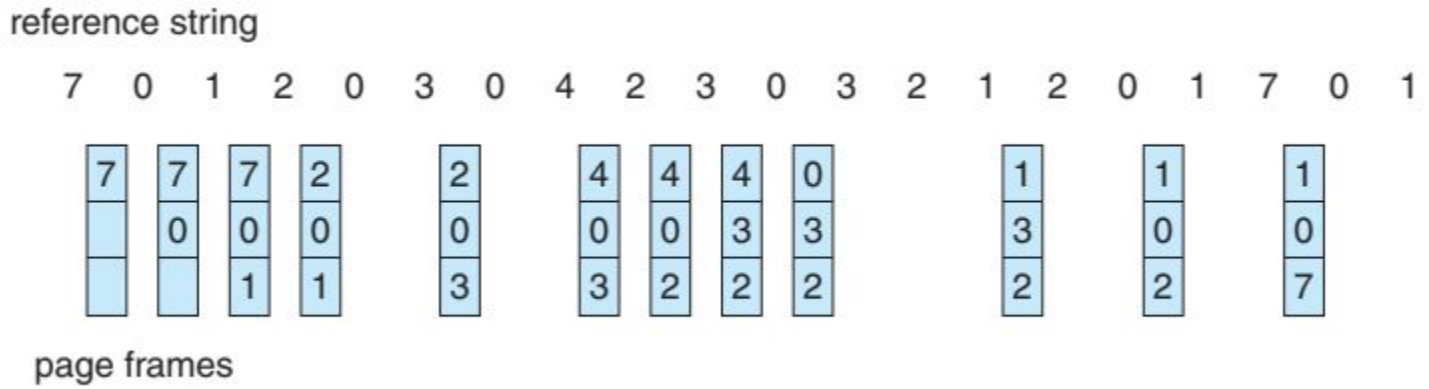
Remanejamento de Páginas

- *Optimal Page Replacement*
  - “Remaneje a página que não será usada pelo maior período de tempo”
  - Difícil de implementar, pois requer conhecimento prévio da *string* de referência;



Remanejamento de Páginas

- *Least Recently Used Algorithm* (LRU)
  - Remaneja a página que **não tem sido usada** pelo maior período de tempo;



Remanejamento de Páginas

- *LRU-Approximation*
  - Alguns sistemas podem fornecer suporte de *hardware* para implementar algoritmos LRU na forma de ***bits* de referência**;
    - Algoritmos de bits de referência adicionais;
    - Algoritmos de Segunda Chance;
    - Algoritmos de Segunda Chance Melhorados;

Remanejamento de Páginas

- *Counting-based*
  - Contador de número de referências feitas para cada página;
    - *Least Frequently Used* (LFU);
    - *Most Frequently Used* (MFU);
- *Page-buffering*
  - *Pool* de frames livres;
  - Lista de páginas modificadas;
  - *Pool* com registro de páginas com seus contadores;

Alocação de frames

- Em um processo, é desejável que haja a menor quantidade de erros de página (*page-fault*) possível. Para tanto, é preciso que haja uma quantidade suficiente de frames para as instruções do processo usarem;
- Estratégias:
  - Mínimo número de frames;
  - Algoritmos de Alocação;

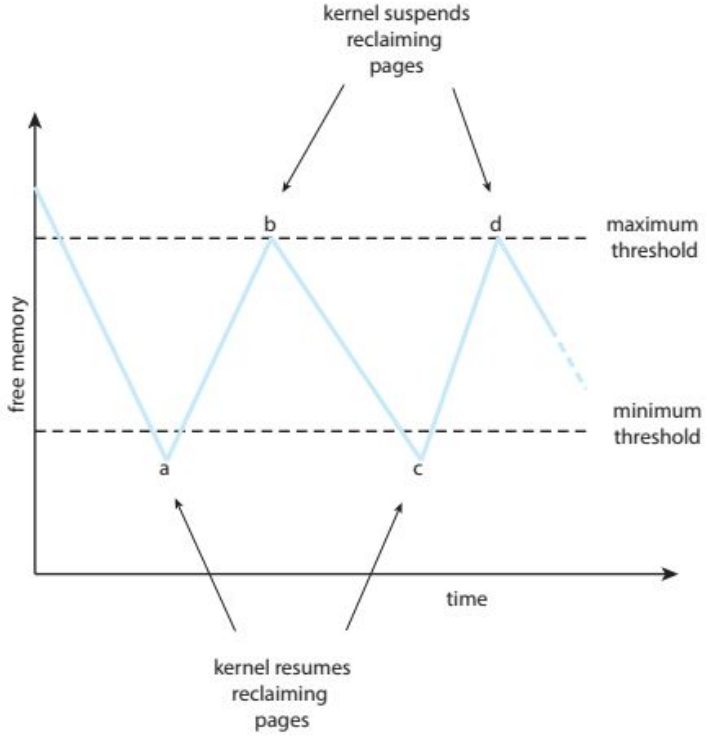
Alocação de frames

- Mínimo número de frames
  - O número mínimo de frames é determinado pela arquitetura do computador;
  - Tem foco na performance;
- Algoritmos de alocação:
  - Alocação igualitária;
  - Alocação proporcional;
  - Em ambos os tipos de alocação, esta pode variar de acordo com o nível de multiprogramação;

Alocação de frames

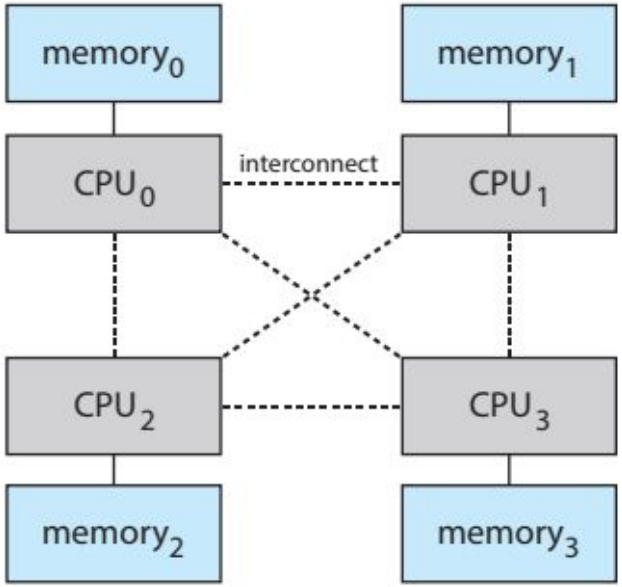
- Alocação global x local:
  - Um problema com algoritmos de alocação global é que o conjunto de páginas na memória para um processo depende do comportamento da página do processo atual e do comportamento das páginas de outros processos;
  - No remanejamento local, o conjunto de páginas na memória para o processo é afetado apenas pelo comportamento da página do processo;
  - O remanejamento global aumenta a produção do sistema (*throughput*);

Alocação de frames



Alocação de frames

- Acesso Não-Uniforme de Memória (NUMA)



Thrashing

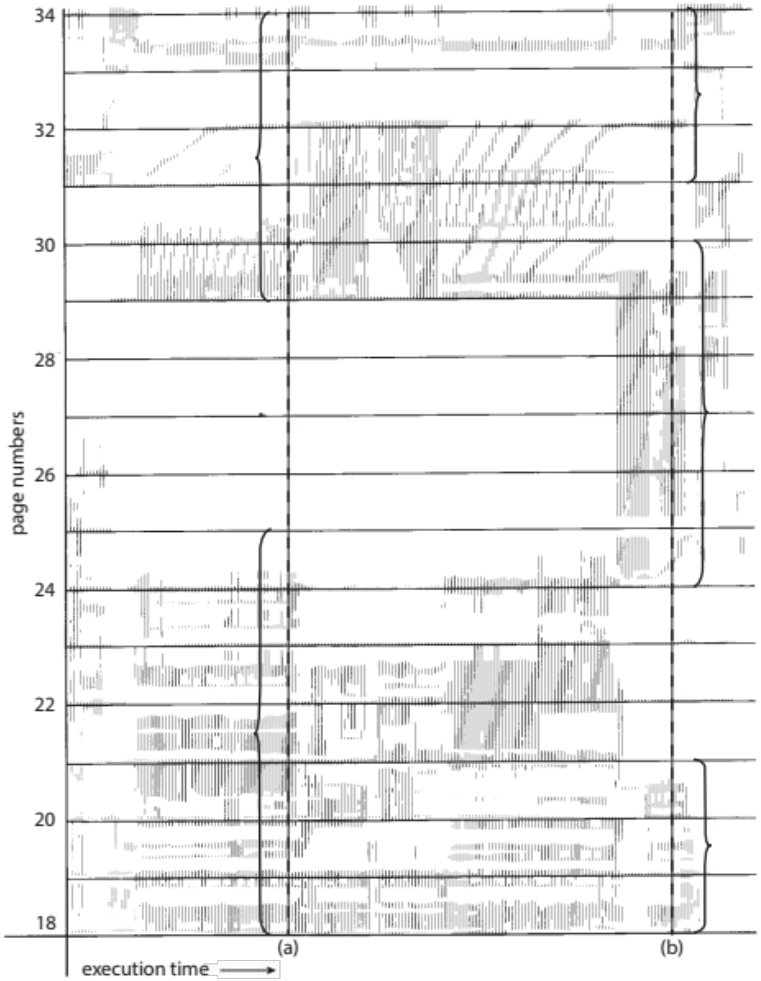
- O fenômeno ocorre quando o grau de multiprogramação é maior do que o grau de utilização do processador;
- O efeito de *trashing* pode ser reduzido com o uso de algoritmos de remanejamento local ou por prioridade;

Thrashing

- Estratégias de prevenção:
  - Modelo de localidade (*locality model*);
  - Modelo de conjunto de trabalho (*working-set model*);
  - Frequência de erros de página (*page-fault frequency*);

Thrashing

- Modelo de localidade
  - Um programa é composto de várias diferentes localidades, as quais podem se sobrepor;
  - Durante a execução, um programa se move de localidade para localidade;
  - As localidades são definidas pela estrutura do programa suas estruturas de dados. O modelo considera que todos os programas vão exibir a mesma estrutura básica de referência de memória;


34
32
30
28
26
24
22
20
18
page numbers
(a)
(b)
execution time

Thrashing

- Modelo do conjunto de trabalho (*working-set model*)
  - Δ (janela do conjunto de trabalho [*working-set window*]);
  - A ideia é examinar as Δ referências de página mais recentes;
  - O conjunto de páginas na mais recente Δ referências de página será o conjunto de trabalho (*working set*);

Thrashing

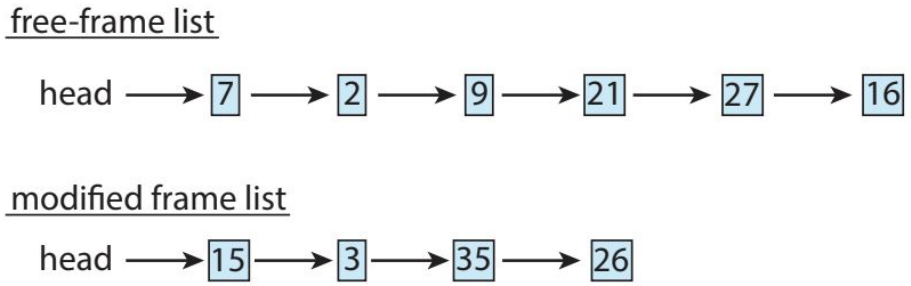
- Modelo do conjunto de trabalho (*working-set model*)
  - A precisão do conjunto de trabalho depende da magnitude de Δ. Um valor baixo não vai abranger toda a localidade, enquanto que um valor alto pode sobrepor várias localidades;
  - O modelo previne o *thrashing* enquanto mantém o grau de multiprogramação tão alto quanto possível. Portanto, ele otimiza a utilização do processador;

Thrashing

- Frequência de erros de página (*page-fault frequency*)
  - Consiste em prevenir o *thrashing* pelo controle da taxa de erros de página;
  - A estratégia é estabelecer limites superiores e inferiores para a taxa de erros desejada.

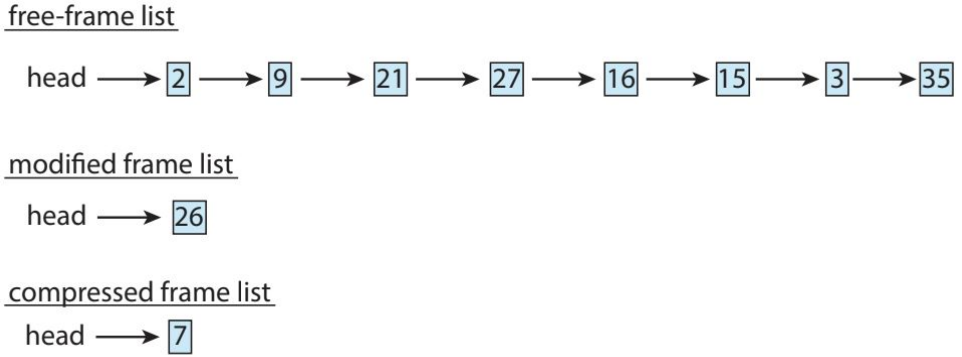
Virtual Memory

- Memory compression
  - Alternativa para paginação;
  - Vários frames em um;
  - Redução de memória.



Virtual Memory

- Memory compression
  - Falha na página;
  - Android, IOS;
  - Windows 10, macOS.



Virtual Memory

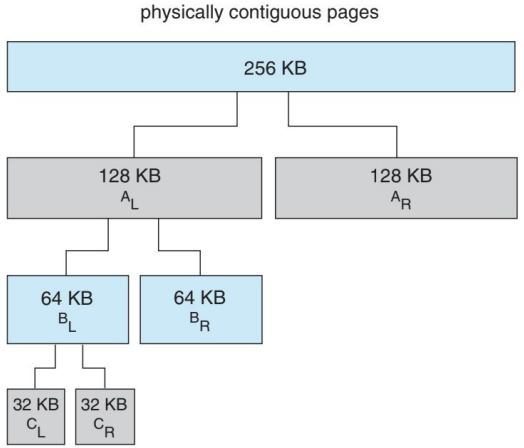
- Memory compression
  - Testes de performance;
  - Algoritmo de compressão (os três frames foram reduzidos à um terço do seu tamanho original);
  - Taxa de compressão (velocidade do algoritmo de compressão e a quantidade de redução atingida).

Allocating Kernel Memory

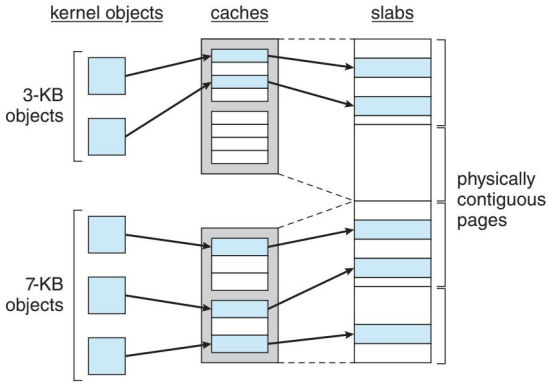
- Memória Kernel
  - Contato direto com o hardware;
  - É uma memória reservada;
  - Fora dos limites do software (off-limits);

Allocating Kernel Memory

- Buddy System
  - Usa o power-of-2 allocator;
  - Pedido em units é arredondado;
  - Coalescing (técnica de combinação).



Allocating Kernel Memory



- Slab Allocation
  - Uma ou mais páginas adjacentes;
  - Eficiente para alocação de memória Kernel;
  - 3 tipos de estados (full(used), empty(free) e partial);
  - Pedidos de memória são respondidos rapidamente;

Outras considerações

- Prepaging
  - Alta quantidade de paginação;
  - Todas as páginas de uma só vez;
- Page size
  - Variação entre 4,906 (2ˆ12) e 4,194,304 (2ˆ22);
  - Tabela de páginas (página virtual e frame na memória);
  - - page size = + número de páginas / + Size tabela de páginas;
  - Fragmentação interna;

Outras considerações

- Alcance TLB
  - Translation lookaside buffer;
  - Se nós aumentarmos o tamanho da página de 4KB para 16KB, nós iremos quadruplicar o alcance do TLB;
  - Fragmentação para aplicações que não precisam de páginas tão grandes.