

Sistemas Operacionais

Cap.8 - Memória Virtual



Prof. MSc. Renzo P. Mesquita
renzo@inatel.br

Objetivos

- Descrever os benefícios de um sistema de Memória Virtual;
- Explicar os conceitos de paginação por demanda, algoritmos de substituição de páginas e alocação de quadros de páginas;



Capítulo 8

Memória Virtual

8.1. Introdução

8.2. Paginação por Demanda

8.3. Cópia-após-Gravação

8.4. Substituição de Páginas

8.5. Alocação de Quadros

8.6. Atividade Improdutiva

8.7. Arquivos Mapeados em Memória



8.1. Introdução

Memória Virtual é uma técnica que permite a execução de processos que não estejam completamente na memória principal.

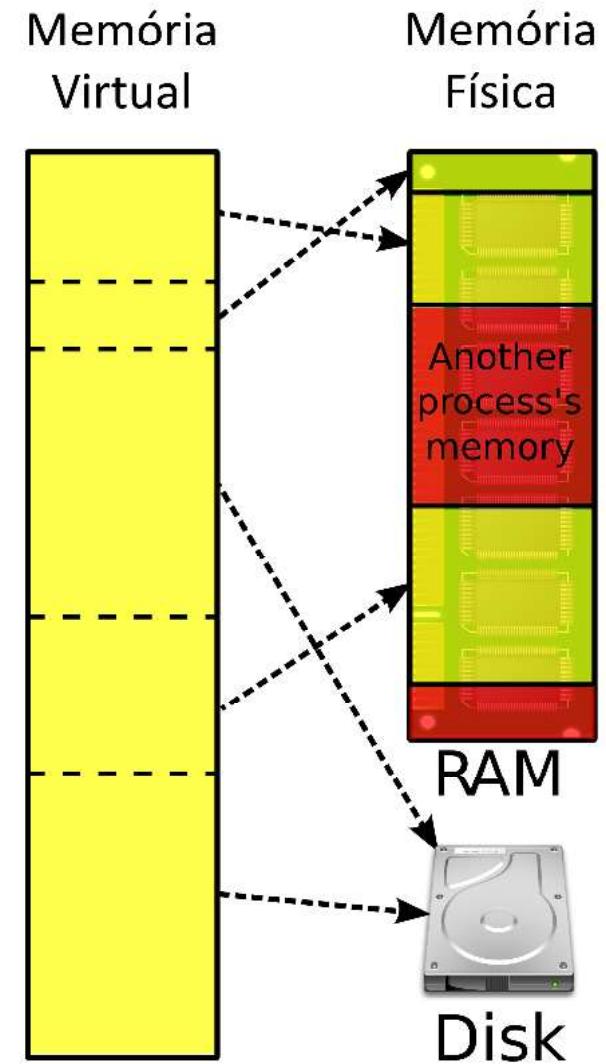
A capacidade de executar um programa que esteja apenas parcialmente em memória poderia trazer muitos benefícios como:

- Um programa não ficaria mais restrito ao **montante de memória física disponível**;
- **Mais programas** poderiam estar **em execução** ao mesmo tempo;
- **Menos I/O** seria necessário para **carregar e remover um programa da memória**;

8.1. Introdução

A Memória Virtual torna a tarefa de programar muito mais fácil, porque o programador não precisa mais se preocupar com a quantidade de memória física disponível;

A Memória Virtual envolve a separação entre a memória lógica como percebida pelos usuários e a memória física. Essa separação permite que uma Memória Virtual extremamente ampla seja fornecida aos programadores quando existe apenas uma Memória Física disponível;



8.2. Paginação por Demanda

8.2.1 Conceitos Básicos

A Paginação por Demanda é uma técnica utilizada em sistemas de memória virtual que permite carregar páginas de Processos somente quando forem necessárias.

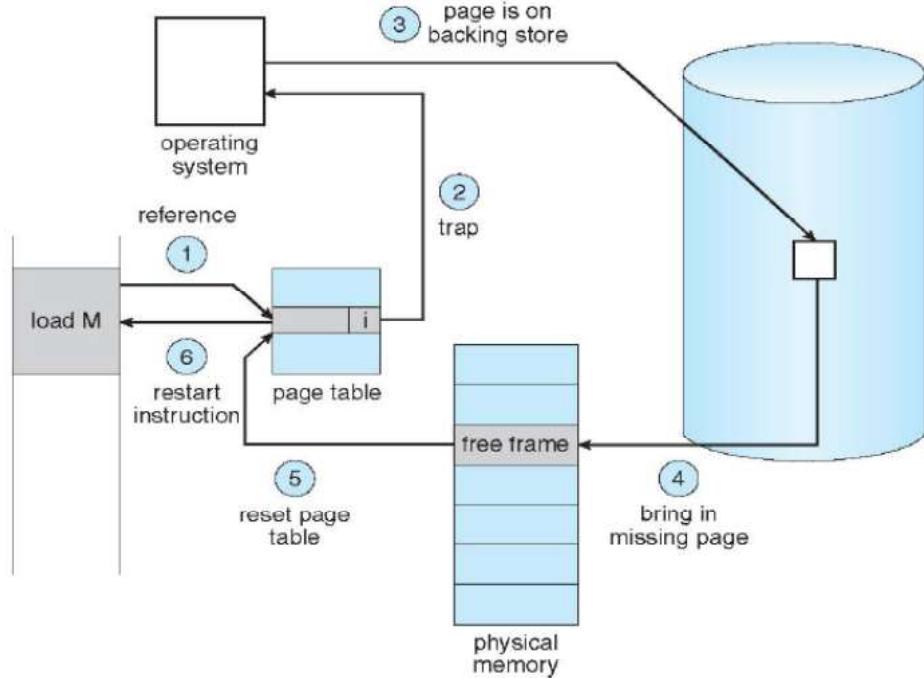
Um sistema que implemente paginação por demanda deve implementar as mesmas estruturas necessárias que um Sistema de Paginação (Page-table, Memória Secundária, etc..) e mais um PAGER;

Um Pager é similar a um Swapper, a diferença é que o Swapper manipula processos inteiros, e o **pager manipula páginas individuais do processo**

Mas o que acontecerá se o processo tentar acessar uma página que ainda não foi carregada em memória?

8.2. Paginação por Demanda

Como vimos, o acesso a uma página marcada como inválida provoca o chamado **Page Fault** (Erro de Página). O procedimento para manipulação deste erro é simples:



1. A tabela de páginas do processo é acessada;
2. Se a página for inválida, uma exceção é causada e o SO é avisado;
3. SO busca página na memória secundária;
4. Aloca um frame (quadro) para a nova página;
5. Atualiza a tabela de Páginas;
6. Reinicia instrução;

8.2. Paginação por Demanda

8.2.2. Observações Importantes

Deve existir um número de páginas mínimas na memória principal para um Processo começar a ser executado?

Quando o SO posicionar o ponteiro de instruções para a primeira instrução do processo provocará imediatamente um Page Fault. O processo continuará sua execução causando Page Faults sempre que necessário até que todas as páginas estejam na memória. Este Esquema é chamado de Paginação por Demanda Pura;

Paginação por Demanda Pura: Jamais carregue uma página em memória até que ela seja solicitada.

E qual seria o tamanho ideal das páginas?

8.2. Paginação por Demanda

Page Size Pequeno (Ex: 512 bytes)

- Diminui a Fragmentação Interna (Internal Fragmentation);
- Aumenta a resolução (Porcentagem de memória ativa);

Page Size Grande (Ex: 8 Kbytes)

- Reduz o overhead devido a redução do número de Page-Faults;
- Reduz o tamanho das tabelas de páginas;

Pelo aumento significativo do tamanho das memórias hoje em dia, a tendência hoje é de se ter page sizes grandes;

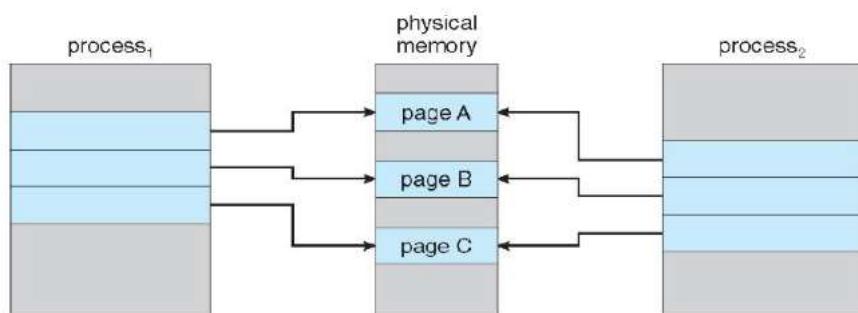
Existem momentos onde é necessário fixar as páginas na memória a fim de que elas não sejam levadas para a memória secundária. Este procedimento é chamado **I/O Interlock**.

Ex: • O Frame esteja passando por uma operação de I/O. Neste caso, caso uma página precise ser liberada com urgência, libera-se a página que foi mapeada mais recentemente;

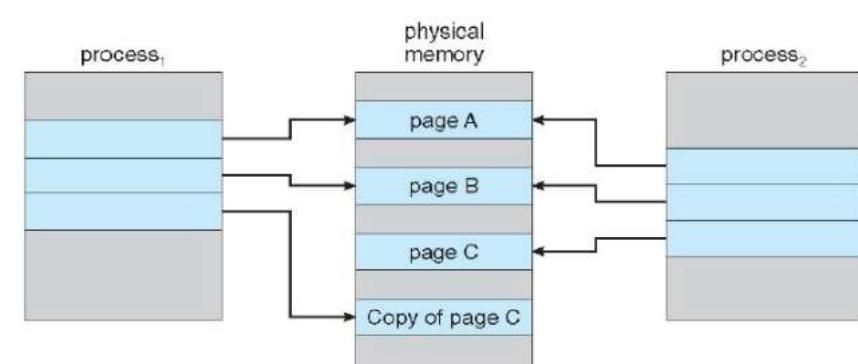
8.3. Cópia-após-Gravação

Cópia-após-gravação é uma técnica que permite que processos Pai e Filho compartilhem inicialmente as mesmas páginas.

- As páginas compartilhadas são marcadas como páginas de cópia-após-gravação, significando que, se um processo gravar em uma página compartilhada, será criada uma nova cópia dessa página.



Antes do Processo 1 modificar a Página C



Após o Processo 1 modificar a Página C

8.4. Substituição de Páginas

Uma Super-alocação ou over-allocating da memória acontece quando ocorre um Page Fault e não há mais frames livres para abrir a nova página.

O que fazer neste caso?

- Pode-se remover um processo qualquer para o disco (**Swapping**) - Método Radical usado quando as taxas de page faults são altas no sistema;
- Pode-se escolher uma Página Vítima (**Victim Page**) e substituí-la pela nova página (Page Replacement) em um Frame de Memória;

Mas como podemos selecionar uma Página Vítima?

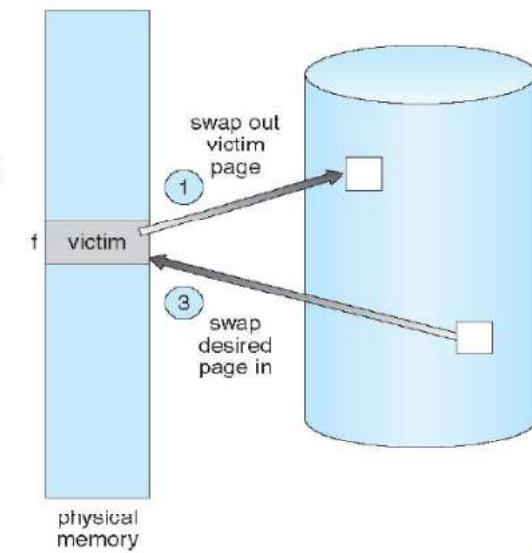
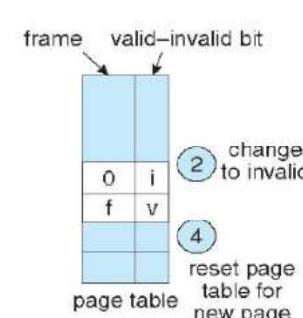
8.4. Substituição de Páginas

Usar um Algoritmo de substituição de páginas para selecionar um Quadro Vítima.

Os Algoritmos mais famosos são:

- *FIFO Algorithm;*
- *OPTIMAL Algorithm;*
- *LRU Algorithm;*
- *Second Chance Algorithm;*
- *LFU Algorithm;*
- *MFU Algorithm;*

Etapas necessárias ao substituir uma Página;



8.4. Substituição de Páginas

8.4.1. FIFO Algorithm

- Algoritmo de substituição de páginas mais simples;
- Associa a cada página a hora em que a página foi carregada na memória;
- Sempre a página mais antiga é a página vítima (página a ser substituída);
- Pode-se também ser criado uma fila para manter todas as páginas em memória. Substituímos a página da cabeça da fila;

Vamos ver um exemplo de como ele funciona?

8.4. Substituição de Páginas

8.4.1. FIFO Algorithm

EXEMPLO 1

Utilize o Algoritmo FIFO para mapear a seguinte sequência de Páginas em uma Memória com 3 Quadros disponíveis: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. Descreva no final quantos Page Faults ocorreram ao analisar esta sequência utilizando deste algoritmo.

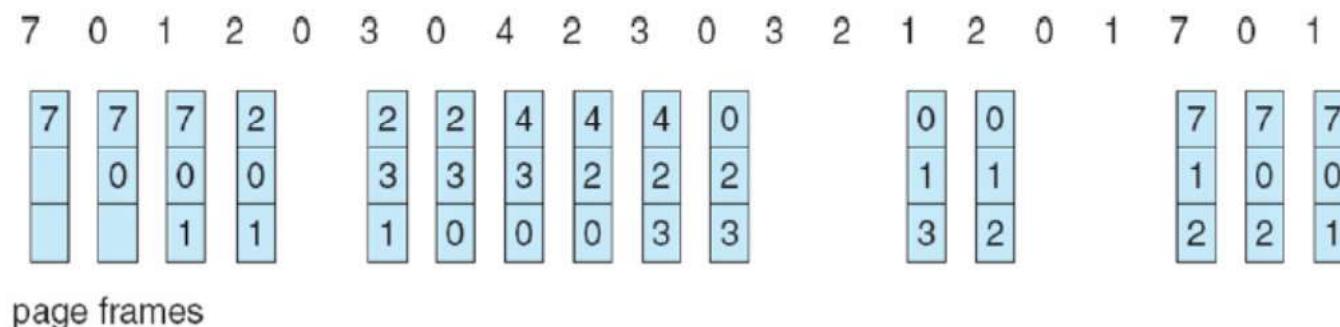


8.4. Substituição de Páginas

8.4.1. FIFO Algorithm

EXEMPLO 1

Utilize o Algoritmo FIFO para mapear a seguinte sequência de Páginas em uma Memória com 3 Quadros disponíveis: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. Descreva no final quantos Page Faults ocorreram ao analisar esta sequência utilizando deste algoritmo.



Total de Page Faults: 15

8.4. Substituição de Páginas

8.4.1. FIFO Algorithm

EXEMPLO 2

Utilize o Algoritmo FIFO para mapear a seguinte sequência de Páginas em uma Memória primeiramente com 3 Quadros disponíveis e depois uma Memória com 4 Quadros disponíveis: 1,2,3,4,1,2,5,1,2,3,4,5. Descreva no final quantos Page Faults ocorreram ao analisar esta sequência utilizando deste algoritmo em cada um dos casos.



8.4. Substituição de Páginas

8.4.1. FIFO Algorithm

No Exemplo 2, algo estranho aconteceu. O Esperado seria que a alocação de mais memória a um processo melhorasse seu desempenho, porém, não foi isso que vimos.

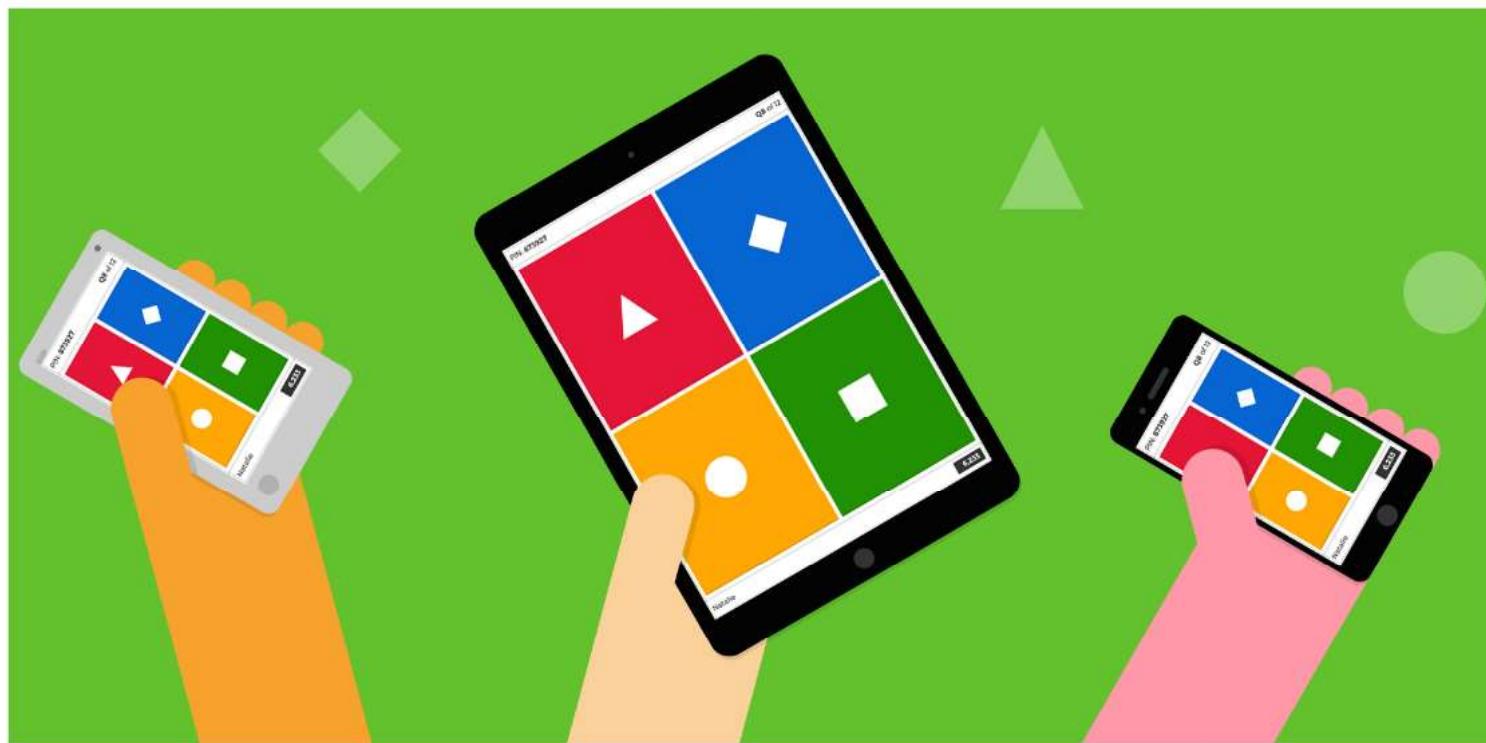
Este resultado tão inesperado é conhecido como **Anomalia de Belady** - Para alguns algoritmos de substituição de páginas, a taxa de Page Faults pode aumentar com o aumento do número de quadros alocados.

Mais Quadros



Mais Page Faults

HORA DO **Kahoot!**



8.4. Substituição de Páginas

8.4.2. *OPTIMAL Algorithm*

- Não sofre da Anomalia de Belady;
- **Algoritmo de substituição de páginas ótimo**, com taxa de erros mais baixa entre todos os Algoritmos;
- Algoritmo de **difícil implementação**, uma vez que **requer o conhecimento futuro** de quais páginas serão referenciadas;
- Funciona da seguinte forma:

Substitua a página que não será utilizada pelo período de tempo mais longo

Vamos ver um exemplo?

8.4. Substituição de Páginas

8.4.2. *OPTIMAL Algorithm*

EXEMPLO 3

Utilize o Algoritmo OPTIMAL para mapear a seguinte sequência de Páginas em uma Memória com 3 Quadros disponíveis: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. Descreva no final quantos Page Faults ocorreram ao analisar esta sequência utilizando deste algoritmo.

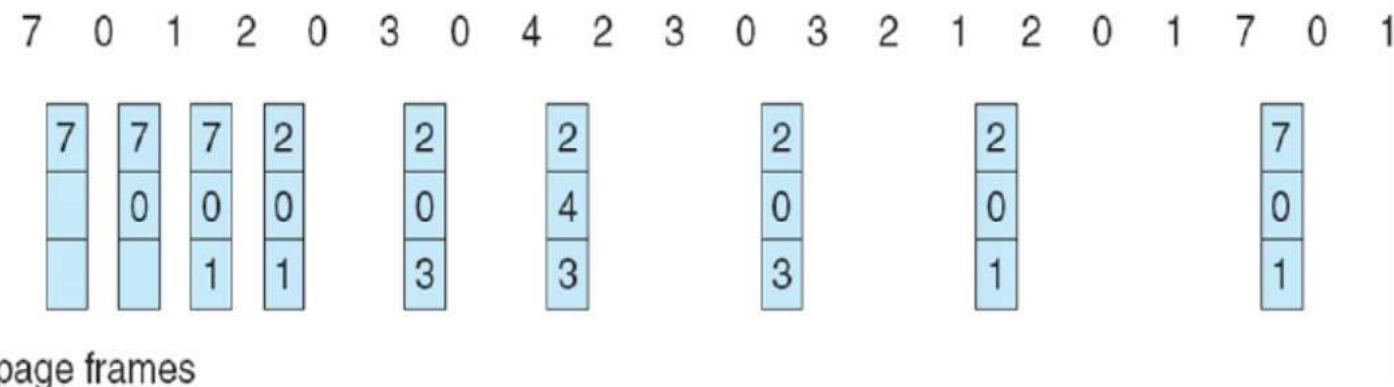


8.4. Substituição de Páginas

8.4.2. *OPTIMAL Algorithm*

EXEMPLO 3

Utilize o Algoritmo OPTIMAL para mapear a seguinte sequência de Páginas em uma Memória com 3 Quadros disponíveis: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. Descreva no final quantos Page Faults ocorreram ao analisar esta sequência utilizando deste algoritmo.



Total de Page Faults: 9

8.4. Substituição de Páginas

8.4.3. LRU (*Least-recently-used*) Algorithm

- FIFO é um Algoritmo que olha para o passado, e OPTIMAL é um algoritmo que olha para o futuro;
- O LRU é um Algoritmo de substituição de páginas ótimo olhando para o passado;
- O LRU escolhe como Página Vítima a página que foi referenciada menos recentemente;
- Também não sofre com a Anomalia de Belady;
- Exige assistência de Hardware, como por exemplo auxílio do TLB;

Vamos ver um exemplo?

8.4. Substituição de Páginas

8.4.3. LRU (Least-recently-used) Algorithm

EXEMPLO 4

Utilize o Algoritmo LRU para mapear a seguinte sequência de Páginas em uma Memória com 3 Quadros disponíveis: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. Descreva no final quantos Page Faults ocorreram ao analisar esta sequência utilizando deste algoritmo.

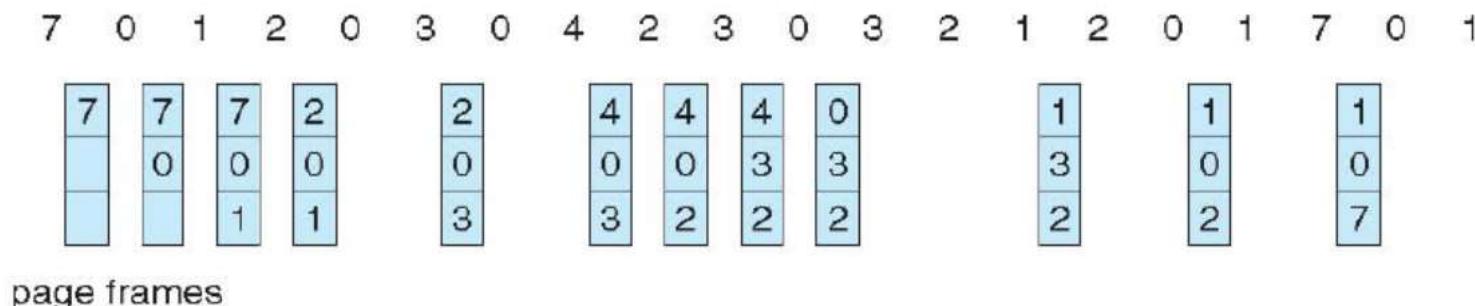


8.4. Substituição de Páginas

8.4.3. LRU (Least-recently-used) Algorithm

EXEMPLO 4

Utilize o Algoritmo LRU para mapear a seguinte sequência de Páginas em uma Memória com 3 Quadros disponíveis: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. Descreva no final quantos Page Faults ocorreram ao analisar esta sequência utilizando deste algoritmo.

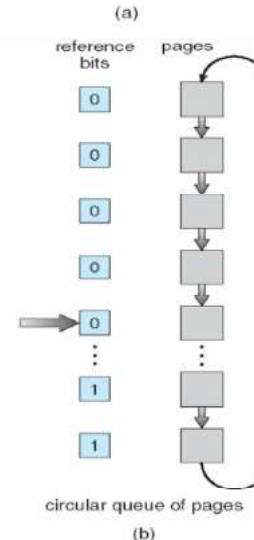
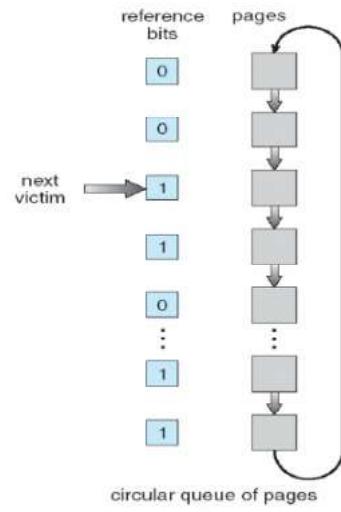


Total de Page Faults: 12

8.4. Substituição de Páginas

8.4.4. Second Chance Algorithm

- Cada página carrega uma informação a mais chamada bit de referência (**reference bit**);
- **Implementa-se o FIFO usual;**
- Quando uma página é selecionada, inspecionamos seu bit de referência. **Se o valor for 0, substituimos essa página; mas se o bit de referência for 1, damos à página uma segunda chance** e seu bit de referência é zerado;
- Uma página que recebe uma segunda chance não será substituída até que todas as outras páginas tenham sido substituídas;
- Implementado como uma Fila Circular;



8.4. Substituição de Páginas

8.4.5. Counting-based Algorithms

Esta classe de Algoritmos implementa um contador do número de referências feitas a cada página. Possui dois Algoritmos usuais:

1) LFU Algorithm

- A página vítima é a página com menor valor no contador;
- Problema: uma página pode ser inicialmente muito usada e depois não mais referenciada. Essa página terá um valor alto no contador e não deixará mais a memória;
- Solução: decrementar todos os contadores em intervalos de tempo regulares;

2) MFU Algorithm

- A página vítima é a página com maior valor no contador;
- Baseia-se no argumento que a página com menor valor no contador foi provavelmente trazida à memória, e portanto, ainda será usada;

Obs: Vale salientar que o custo de implementação do LFU e do MFU é alto, e ambos não se aproximam do desempenho do Optimal Algorithm;

8.5. Alocação de Quadros

A questão agora é: como dividir a quantidade de memória livre (quadros) entre os vários processos?

Pode haver um número mínimo de frames alocados a cada processo e esse número mínimo pode permitir a execução de qualquer instrução do processador sem que haja Page Fault.

Pode-se usar Algoritmos de alocação para realizarem esta distribuição, são eles:

1) Equal Allocation (Alocação Igual)

A maneira mais fácil de dividir m quadros entre n processos dando a cada processo uma parte igual correspondente a m/n quadros;

2) Proportional Allocation (Alocação Proporcional)

A memória disponível é alocada a cada processo de acordo com seu tamanho;

3) Priority Allocation (Alocação com Prioridade)

A memória disponível é alocada a cada processo de acordo com a sua prioridade;

8.5. Alocação de Quadros

Outro fator importante que influí na maneira como os quadros são alocados aos diversos processos é a substituição de páginas.

Com múltiplos processos competindo por quadros, o SO pode realizar a substituição de páginas de duas formas diferentes, são elas:

1) Substituição Global (Global Replacement)

Permite a um processo obter um quadro mesmo quando este se encontra alocado a um outro processo; Este método é o mais utilizado.

2) Substituição Local (Local Replacement)

Permite a um processo obter um quadro às custas de ter que liberar algum outro de seus próprios frames;

8.6. Atividade Improdutiva (Thrashing)

Se um Processo tem um número muito pequeno de frames (quadros), ele entra em page faults sucessivos, num estado chamado de **Thrashing**.

Um Processo que se encontra em Thrashing, ele gasta mais tempo paginando do que executando.

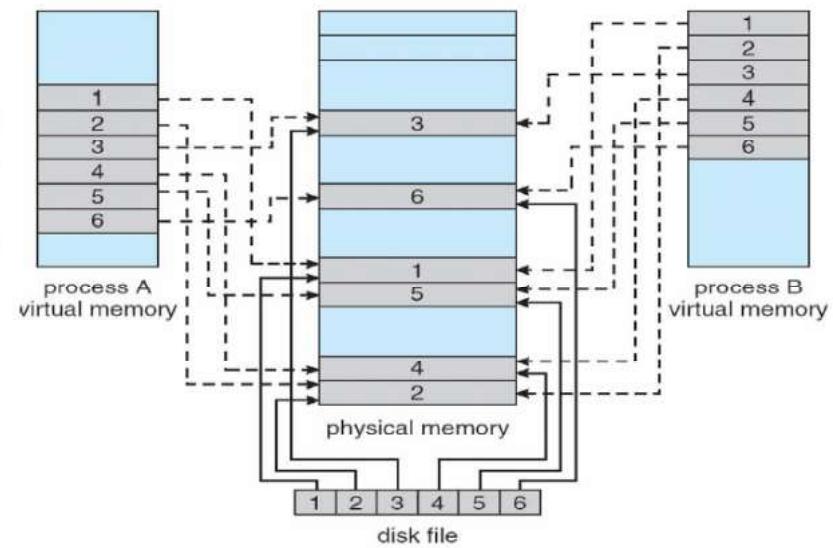
No nível mais básico, para evitar o Thrashing de Processos, pode-se fazer o seguinte:

- Aumentar a quantidade de RAM de um computador;
 - Diminuir o número de programas rodando no computador;
 - Utilizar programas que utilizem menos memória;
 - Setar prioridades aos programas;
- entre outros..

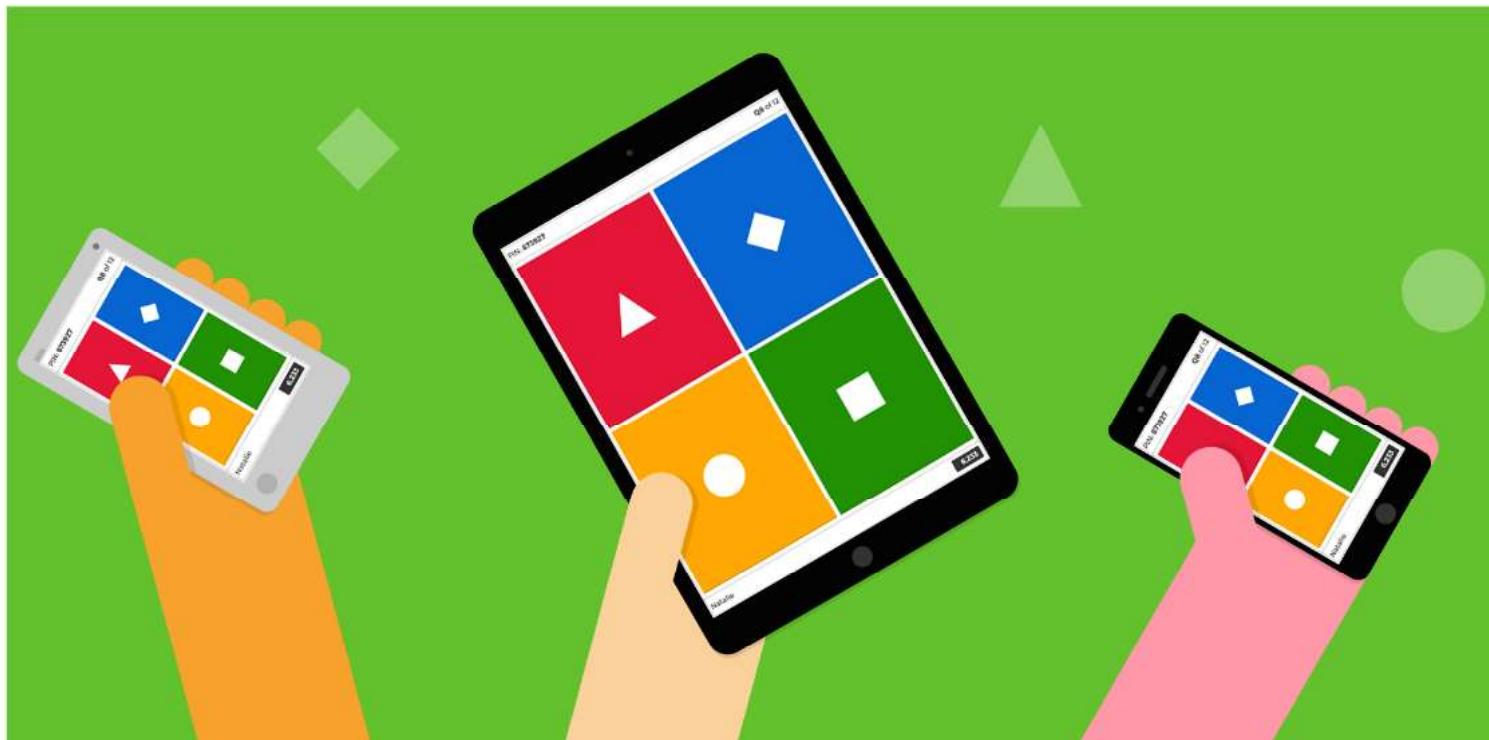
8.7. Arquivos Mapeados em Memória

Mapear um arquivo em memória permite que um bloco do arquivo em disco seja mapeado em uma ou mais páginas do processo em memória.

- Quando o processo executa a **primeira operação de read** no arquivo, **blocos do arquivo são trazidos para uma ou mais páginas do processo em memória**;
- A partir daí, as **demais operações de read (ou write)** são realizadas diretamente na(s) página(s) mapeadas em memória;
- Isso leva a uma **melhora significativa de desempenho** na execução de I/O;
- Vários Processos podem usufruir deste arquivo na memória caso tenham permissão;



HORA DO **Kahoot!**



**FIM
DO
CAPÍTULO 8**



EXERCÍCIOS