

Sistemas Operacionais

Gerencia de Memória

Sumário

1 Gerencia de Memória

2 Partições

3 Swapping

4 Paginação

5 Seguimentação

6 Seguimentação Páginaada

7 Bibliografia

Gerencia de Memória

Na multiprogramação, vários processos são executados ao mesmo tempo por meio da divisão do tempo do processador. Para que a troca entre eles seja rápida, os processos devem estar na memória, prontos para execução. Cabe à gerência de memória do sistema operacional fornecer os mecanismos para que esses processos compartilhem a memória com segurança e eficiência.

(OLIVEIRA; CARISSIMI; TOSCANI, 2001)

Conceito geral

A gerência de memória é responsável por prover mecanismos que permitam aos processos compartilhar a memória de forma segura e eficiente.

- › Processos devem estar carregados na memória para execução.
- › O sistema operacional organiza e protege o acesso à memória.
- › As técnicas variam conforme o suporte do *hardware*.

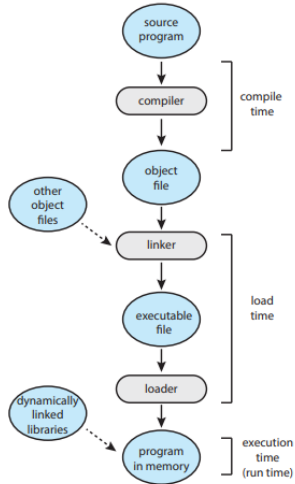


Figura 1: Programa para processo (SILBERSCHATZ; GALVIN; GAGNE, 2001).

- **Memória física:** É aquela implementada pelos circuitos integrados de memória, pela eletrônica do computador. O endereço físico é aquele que vai para a memória física, ou seja, é usado para endereçar os circuitos integrados de memória.
- **Memória lógica:** O espaço de endereçamento lógico de um processo é formado por todos os endereços lógicos que esse processo pode gerar. Existe um espaço de endereçamento lógico por processo. Já o espaço de endereçamento físico é formado por todos os endereços aceitos pelos circuitos integrados de memória.

Memória Lógica e Física II

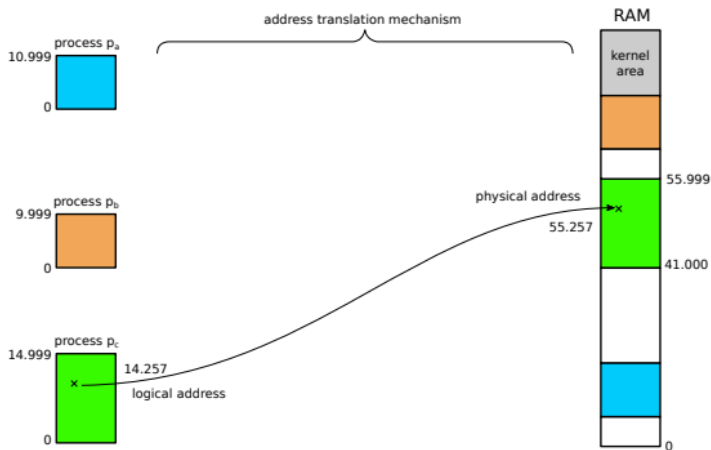


Figura 2: Memória física e virtual (Maziero, 2019).

Endereçamento e MMU

- Cada processo possui sua própria memória lógica.
- Endereços lógicos são convertidos em endereços físicos pela MMU (*Memory Management Unit*).
- A proteção pode ser realizada por registradores de base e limites.

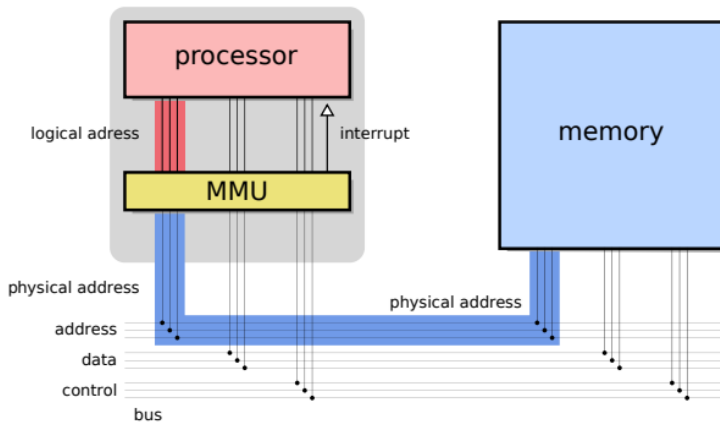


Figura 3: MMU entre processador e memória (Maziero, 2019).

- › Registradores de limite inferior/superior e base/limite devem ser protegidos e não acessíveis em modo usuário.
- › Devem ser acessíveis apenas em modo supervisor.
- › Seus valores integram o contexto de execução dos processos.
- › Podem ser armazenados no descritor de processo.
- › Durante o chaveamento de processo, os valores são copiados do DP para os registradores da MMU.
- › Isso limita a região de memória acessível ao processo que recebe o processador.

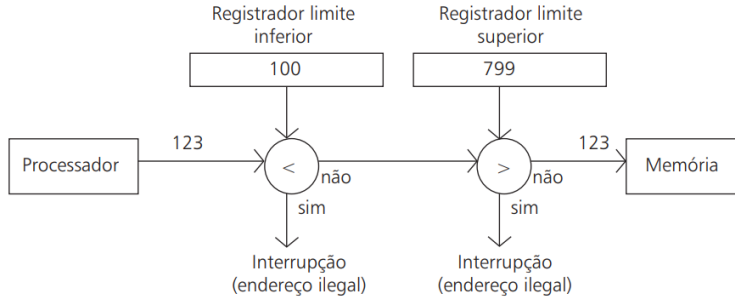


Figura 4: Carregador Relocador (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

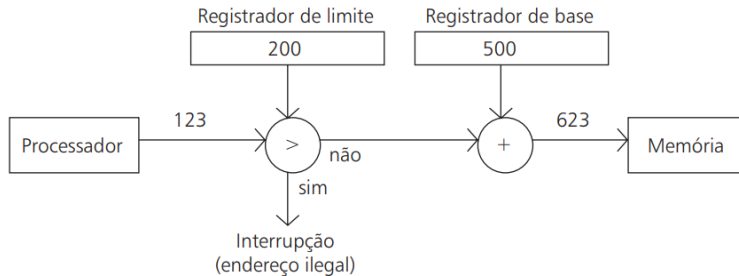


Figura 5: Carregador de Absoluto (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

- › Existem duas abordagens comuns para proteger e relocar programas na memória: com **registradores de limite** e com **registradores de base e limite**.
- › **Esquema com registradores de limite:**
 - › Os programas são gerados para iniciar no endereço lógico zero.
 - › Na prática, são carregados em posições físicas diferentes, de acordo com a memória disponível no momento.
 - › Para funcionar corretamente, os endereços do programa precisam ser ajustados durante a **carga** — esse processo é chamado de **relocação**.
 - › O componente responsável por isso é o **carregador relocador**, que modifica os endereços para refletir a posição real ocupada.

› Esquema com registradores de base:

- › Os programas também são gerados para iniciar no endereço lógico zero.
- › Podem ser carregados em qualquer lugar da memória física, sem necessidade de alterar seus endereços.
- › O valor do **registrador de base** é somado automaticamente aos endereços lógicos, produzindo os endereços físicos corretos.
- › Nesse caso, não há necessidade de modificar o programa durante a carga, pois a relocação acontece dinamicamente em **tempo de execução**.
- › Por isso, o carregador usado é chamado de **carregador absoluto**.

- A relocação em tempo de carga ou em tempo de execução depende diretamente do suporte oferecido pelo **hardware**.
- Esse suporte é fornecido pela **MMU (Unidade de Gerência de Memória)**, que permite ao sistema operacional implementar mecanismos eficientes de proteção e alocação de memória.

Partições

A memória é dividida em partições fixas.

› Problemas:

- ›› **Fragmentação interna:** ocorre quando sobra memória *dentro* da partição ocupada, pois o processo não utiliza todo o espaço alocado.
- ›› **Fragmentação externa:** ocorre quando há memória livre *fora* das partições, mas dividida em blocos pequenos e não contíguos, impedindo o uso eficiente.

Memória Física

Sistema Operacional - 225 <i>Kbytes</i>
Partição 1 - 200 <i>Kbytes</i>
Partição 2 - 100 <i>Kbytes</i>
Partição 3 - 50 <i>Kbytes</i>
Partição 4 - 25 <i>Kbytes</i>

Figura 6: Exemplo de partições fixas (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

Partição Variáveis

Quando partições variáveis são empregadas, o tamanho das partições é ajustado dinamicamente às necessidades exatas dos processos. Essa é uma técnica de gerência de memória mais flexível que partições fixas.

- Existem quatro algoritmos para percorrer a lista de lacunas:
 - **first-fit**: Utiliza a primeira lacuna que encontrar com tamanho suficiente;
 - **best-fit**: Utiliza a lacuna que resultar na menor sobra;
 - **worst-fit**: Utiliza a lacuna que resultar na maior sobra;
 - **circular-fit**: Como *first-fit*, mas inicia a procura na lacuna seguinte à última sobra.
- Fragmentação externa continua sendo um problema.

Partições Variáveis II

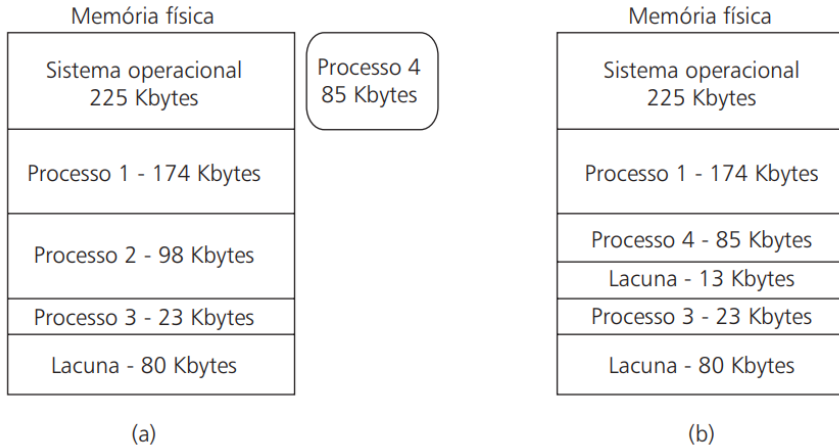


Figura 7: Partições variáveis (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

Parágrafos:

- Em alguns sistemas, a memória é organizada em unidades chamadas **parágrafos**, geralmente com tamanho fixo de 32 bytes.
- Nesse esquema, toda alocação de memória deve ser um múltiplo inteiro de parágrafos.
- Assim, mesmo que um processo precise de 45 bytes, ele receberá 64 bytes (2 parágrafos), o que pode gerar até **31 bytes de fragmentação interna** por processo.

Compactação:

- **Fragmentação externa** é um problema sério em partições variáveis: ao alocar e liberar memória, surgem muitas lacunas pequenas demais para serem úteis.
- A memória muitos espaços desperdiçados entre os processos.
- Estima-se que até **1/3 da memória** possa ser desperdiçada devido à fragmentação externa.
- Uma possível solução é a **compactação de memória**, que move os processos para juntar as lacunas livres.
- No entanto, esse processo é lento, exige muito do processador e depende de **relocação dinâmica** (como via registrador de base).
- Por isso, a compactação geralmente **não é utilizada na prática**.

Swapping

O que é?

Quando não há memória suficiente para manter todos os processos ativos, o sistema pode usar o mecanismo de **swapping**, que consiste em transferir processos entre a memória principal e o disco.

Funcionamento do swapping:

- Um processo é **copiado da memória para o disco** (swap-out), sendo suspenso temporariamente.
- Mais tarde, ele pode ser **carregado novamente para a memória** (swap-in), retomando sua execução.
- Com isso, o sistema consegue executar mais processos do que caberiam simultaneamente na memória.

Situações típicas de uso:

- Quando um processo solicita mais memória e não há espaço contíguo suficiente.
- Quando um novo programa é iniciado, e não há memória disponível no momento.

Custo do swapping:

- É uma operação demorada: exige copiar todo o processo entre a memória e o disco.
- Só é justificável se o processo ficar tempo suficiente fora da memória.

Quando o swapping é mais adequado:

- **Não recomendado** em ambientes interativos (ex: terminais), pois o usuário sente a lentidão.
- **Mais aceitável** em processos em segundo plano (*background*), sem interação direta.

Correção de endereços:

- Se o processo voltar para um local diferente da memória, seus endereços precisam ser corrigidos.
- Com uso de **registrador de base**, basta atualizar o valor do registrador — não é necessário alterar o código do processo.

Paginação

Eliminação da fragmentação externa

- A técnica de **partições fixas** causa alto desperdício de memória — por isso, caiu em desuso.
- As **partições variáveis** são mais flexíveis, mas ainda sofrem com **fragmentação externa**.
- Essa fragmentação ocorre porque os programas precisam ocupar uma área **contígua** na memória.
- Se permitirmos que um programa seja carregado em **várias áreas não contíguas**, eliminamos esse problema.
- A técnica que permite essa divisão e elimina a fragmentação externa é chamada de **paginação**.

- › Memória dividida em páginas (fixas).
- › Programa pode ocupar páginas não contíguas.
- › Fragmentação interna ainda ocorre.

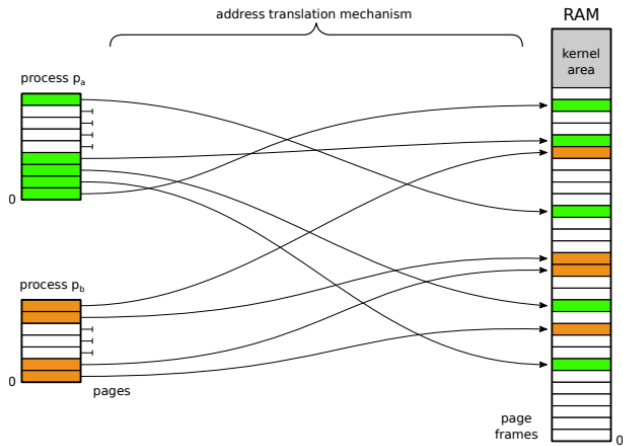


Figura 8: Exemplo de paginação (Maziero, 2019).

- › Durante a execução, o processo gera **endereços lógicos** para acessar memória.
- › O programa assume que está em uma área contígua, iniciando no endereço lógico zero.
- › Para funcionar corretamente, é necessário **converter esse endereço lógico em físico**.
- › Essa conversão é feita com ajuda da **tabela de páginas** mantida pelo sistema operacional.
- › O endereço lógico é dividido em duas partes:
 - » **Número da página lógica** (índice na tabela de páginas);
 - » **Deslocamento** dentro da página.
- › A entrada da tabela fornece o número da **página física** correspondente.
- › O endereço físico final é formado ao combinar a página física com o deslocamento.

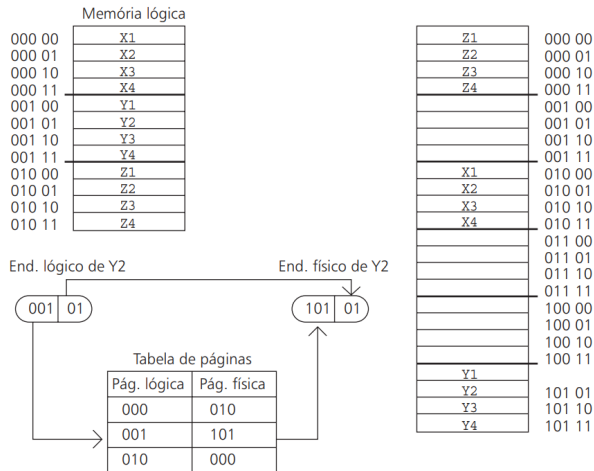


Figura 9: Exemplo de paginação (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

- › Vantagens de utilizar páginas grandes:
 - » O processo terá menos páginas.
 - » A tabela de páginas será menor.
 - » A leitura do disco será mais eficiente.
 - » Em geral, há um custo menor imposto pelo mecanismo de gerência de memória (*overhead* reduzido).
- › Desvantagens de utilizar páginas grandes:
 - » Maior fragmentação interna.

 Observação:

O tamanho das páginas é fixado pelo hardware, especificamente pela Unidade de Gerência de Memória (MMU).

› Tabela de páginas pequena:

- ›› Pode ser armazenada completamente em registradores rápidos.
- ›› Registradores proporcionam acesso rápido sem degradar o tempo de acesso à memória.
- ›› No chaveamento de processos, a tabela de páginas do processo ativo é copiada do descritor de processo (DP) para os registradores.

› Tabela de páginas grande:

- ›› Não cabe em registradores, então é armazenada na memória principal.
- ›› A MMU usa dois registradores:
 - PTBR (Page Table Base Register): aponta para o endereço físico da tabela.
 - PTLR (Page Table Limit Register): indica o número de entradas da tabela.
- ›› Cada acesso à memória lógica envolve dois acessos à memória física:
 - Primeiro: consulta à tabela de páginas para traduzir o endereço.
 - Segundo: acesso ao dado na memória física.
- ›› No chaveamento de processos, PTBR e PTLR são atualizados com os valores do DP.

› Uso do Translation Lookaside Buffer (TLB):

- › Memória cache especial que armazena as entradas da tabela de páginas mais recentemente usadas.
- › Acesso rápido, reduzindo o número de acessos à memória física.
- › *Hit* na TLB: acesso à memória é feito em um único passo.
- › *Miss* na TLB: são necessários dois acessos, e a entrada é carregada na TLB para acessos futuros.
- › Taxas de acerto típicas: 80% a 90%, reduzindo o overhead da paginação para apenas 20% a 30%.

› Chaveamento de processos com TLB:

- › PTBR e PTLR devem ser atualizados no chaveamento.
- › A TLB deve ser esvaziada (flushed) para evitar usar entradas de processos anteriores.

Seguimentação

Organização por significado

- › Divisão lógica em segmentos: código, dados, pilha, etc.
- › Facilita compartilhamento e proteção.
- › Alocação contígua leva à fragmentação externa.

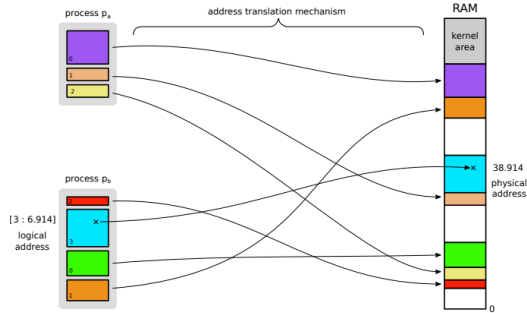


Figura 10: Gerência baseada em segmentos (Maziero, 2019).

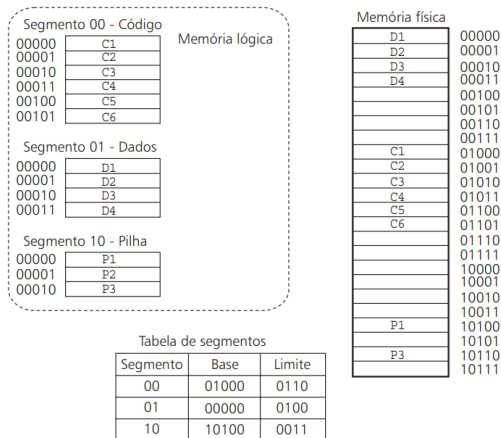


Figura 11: Gerência baseada em segmentos (Maziero, 2019).

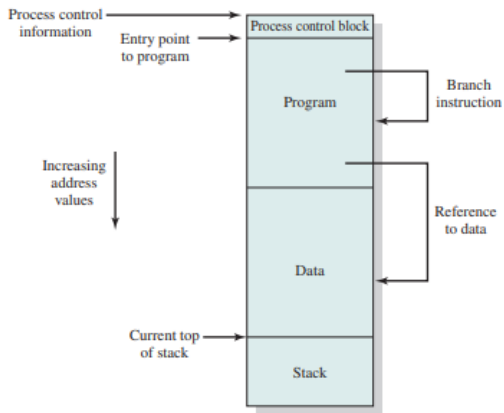


Figura 12: Seguimentação de um processo (STALLINGS, 2009).

› Página:

- ›› Conceito criado pelo **sistema operacional**.
- ›› Utilizado para facilitar a **gerência da memória**.

› Visão do programador/compilador:

- ›› Não enxergam a memória como páginas.
- ›› Utilizam o conceito de **segmentos** lógicos.

› Segmentação típica de um programa:

- ›› **Código**: instruções do programa.
- ›› **Dados estáticos**: variáveis alocadas estaticamente (ex.: globais).
- ›› **Dados dinâmicos**: variáveis criadas em tempo de execução (ex.: com `malloc`).
- ›› **Pilha**: usada para chamadas de funções e variáveis locais.

Seguimentação Página da

- Cada segmento é dividido em páginas.
- Elimina fragmentação externa, mas introduz interna.
- Tabelas de segmentos apontam para tabelas de páginas.

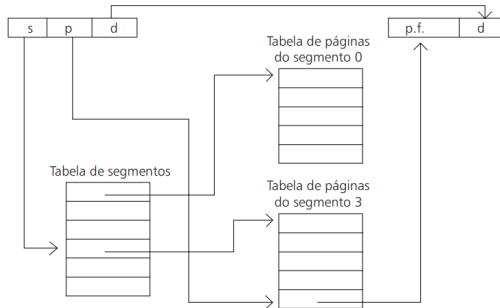


Figura 13: Segmentação paginada (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

Capítulo 6 (OLIVEIRA; CARISSIMI; TOSCANI, 2001)



Bibliografia



OLIVEIRA, R. S.; CARISSIMI, A. S.; TOSCANI, S. S. **Sistemas Operacionais**. 2. ed. Porto Alegre: Sagra-Luzzatto, 2001.



STALLINGS, William. **Operating Systems: Internals and Design Principles**. 6. ed. Upper Saddle River, NJ: Prentice-Hall, 2009.



TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2010.



SILBERSCHATZ, Abraham; GALVIN, Peter; GAGNE, Greg. **Sistemas Operacionais: Conceitos e Aplicações**. Rio de Janeiro: LTC, 2001.

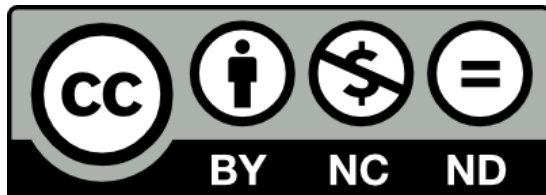


TANENBAUM, Andrew S.; WOODHULL, Albert S. **Sistemas Operacionais: Projeto e Implementação**. 2. ed. Porto Alegre: Bookman, 2000.



MAZIERO, Carlos Alberto. **Sistemas Operacionais: Conceitos e Mecanismos [recurso eletrônico]**. Curitiba: DINF - UFPR, 2019. ISBN 978-85-7335-340-2.

Estes slides estão protegidos por uma licença Creative Commons



Este modelo foi adaptado de Maxime Chupin.

Sistemas Operacionais

Gerencia de Memória