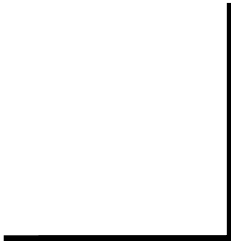


**Universidade Federal do Rio Grande do Norte
Unidade Acadêmica Especializada em Ciências Agrárias
Escola Agrícola de Jundiaí
Curso de Análise e Desenvolvimento de Sistemas
TAD0006 - Sistemas Operacionais - Turma 01**

Gerenciamento de Memória

Antonino Feitosa
antonino.feitosa@ufrn.br

Macaíba, maio de 2025



Aula Passada

- Escalonamento e Algoritmo de Escalonamento
- Processo CPU-bound e IO-bound
- Quando ocorre o escalonamento
- Custo do escalonamento
- Objetivos dos algoritmos de escalonamento
- Política versus Mecanismo

Aula Passada

- Algoritmos
 - Primeiro a Chegar, Primeiro a Ser Servido
 - Tarefa mais Curta Primeiro
 - Tempo mais Curto Primeiro
 - Chaveamento Circular
 - Prioridades
 - Processo mais Curto Primeiro

Roteiro

- Introdução
- Espaço de Endereçamento
- Swapping
- Memória Virtual

Introdução



Introdução

- Memória principal (RAM): um dos principais componentes de um sistema computacional.
 - Atualmente armazenam uma quantidade razoável de dados.
 - Lei de Parkinson: "programas tendem a expandir-se a fim de preencher a memória disponível para contê-los".
- Como gerenciar a memória?
- Quais as abstrações estão envolvidas neste gerenciamento?

Introdução

- Desejamos uma memória privada, infinita, rápida, não volátil e barata.
 - Até o momento, ela não foi desenvolvida!
- Hierarquia de Memória:
 - Cache: volátil, pequena e rápida.
 - RAM: volátil, tamanho e velocidade razoáveis.
 - Memória Secundária: não volátil, grande e lenta.
 - HD, SSD e outros dispositivos como pen drive, entre outros.
- O sistema operacional deve abstrair a hierarquia em um modelo útil e então gerenciar essa abstração.
 - Gerenciador de memória: parte do SO que gerencia a hierarquia de memória.

Introdução

- Observe que a memória cache é gerenciada pelo hardware.
- Foco no gerenciamento da memória principal (RAM).
- Iniciaremos pelo modelo mais simples: sem abstrações.

Sem Abstração de Memória

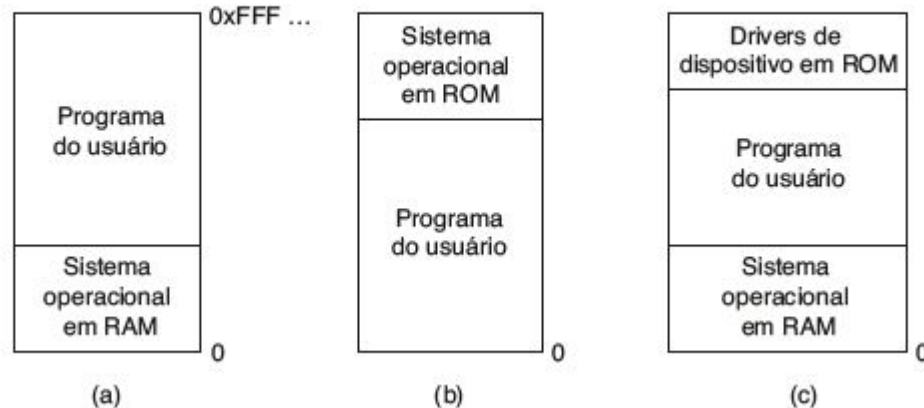
Sem Abstração de Memória

- Os programas manipulam a memória física diretamente.
- Não é possível executar dois programas ao mesmo tempo.
 - Os processo referenciam os endereços físicos diretamente, que podem ser os mesmos de outro processo.
 - O que ocorre com a alocação estática?
- Não é possível garantir segurança.
 - Qualquer processo pode alterar qualquer endereço na memória RAM, comprometendo o funcionamento dos demais.
- A organização da memória varia quanto à porção dedicada ao sistema operacional.

Sem Abstração de Memória

- O que ocorre se um programa alterar uma posição indevida pertencente ao sistema operacional?

FIGURA 3.1 Três maneiras simples de organizar a memória com um sistema operacional e um processo de usuário. Também existem outras possibilidades.



Sem Abstração de Memória

- O modelo b) geralmente é usado em sistemas embarcados.
- Podemos fazer o uso de threads para simular paralelismo.
 - Limitado por não fornecer processos.
 - Em um sistema tão simples (sem abstração de memória) é improvável que forneçam suporte à threads.

Sem Abstração de Memória

- Como executar múltiplos programas?
 - Podemos salvar o conteúdo inteiro da memória em um arquivo de disco, então introduzir e executar o programa seguinte.
 - Swapping
 - Não há conflitos se apenas um processo estiver na RAM.
- Podemos utilizar hardware especializado.

Sem Abstração de Memória

- Apesar de simples e limitada ela ainda é usada em sistemas embarcados, de cartões inteligentes e em memórias ROM que utilizam endereçamento absoluto.

Espaços de Endereçamento



Espaços de Endereçamento

- Problemas em expor a memória física:
 - Podem corromper o sistema operacional de modo, intencionalmente ou por acidente.
 - Acesso a qualquer endereço.
 - Proteção!
 - Difícil mantermos dois programas na memória.
 - Endereços compartilhados.
 - Realocação!

Espaços de Endereçamento

- Necessitamos de proteção e realocação para mantermos múltiplos processos na memória.
- Espaço de endereçamento: é o conjunto de endereços que um processo pode usar para endereçar a memória.
 - Memória abstrata para execução dos programas.
 - Cada programa possui sua memória dedicada (abstrata).
 - Cada processo possui seu espaço de endereçamento, independente dos demais processos.

Espaços de Endereçamento

- Essa abstração ocorre em diferentes contextos:
 - Números de telefone: cada número identifica uma conta.
 - Todos os valores de 00000000 até 999999999.
 - Endereços de portas de E/S.
 - Todos os valores de 0 até um valor máximo 2^{16} , 2^{32} , etc.
 - Domínios da internet.
 - Todas as cadeias de caracteres de tamanho 2 até 64, formados por letras e dígitos.

Espaços de Endereçamento

- Como determinar um espaço de endereçamento para cada processo?
- Por exemplo, desejamos garantir que o endereço 28 de processo corresponda a um endereço físico diferente do endereço 28 de outro processo.
- Solução simples: realocação dinâmica.

Realocação Dinâmica

Realocação Dinâmica

- Mapeamento do espaço de endereçamento de cada processo em uma parte diferente da memória física.
 - Os programas são carregados em posições de memória consecutivas sempre que haja espaço.
 - Use de dois registradores de controle:
 - Registrador base: armazena o endereço físico onde o programa começa.
 - Registrador limite: armazena o comprimento do programa.

Realocação Dinâmica

- Ao acessar um endereço X , o valor do registrador base é somado ao X antes de enviá-lo para o barramento de memória.
- Também é verificado se X está dentro do programa.
 - Menor ou igual ao valor do registrador limite.
 - Caso contrário, é gerada uma falta e o acesso é abortado.

Realocação Dinâmica

- Modo simples de fornecer espaços de endereçamentos para cada processo de modo independente.
 - Garante proteção.
 - Garante realocação.
 - Os registradores base e limite devem ser protegidos pelo SO.
- Desvantagem de exigir uma **adição** e uma **comparação** a cada acesso à memória.

Swapping



Swapping

- Na prática, a memória RAM não é grande o suficiente para manter todos os processos em execução.
- Processos ociosos podem ser armazenados em disco em sua maior parte, sendo transferidos para memória RAM ao despertarem, e então voltam a dormir.
- Como permitir que todos esses processos consigam executar?
 - Compartilhamento da memória.

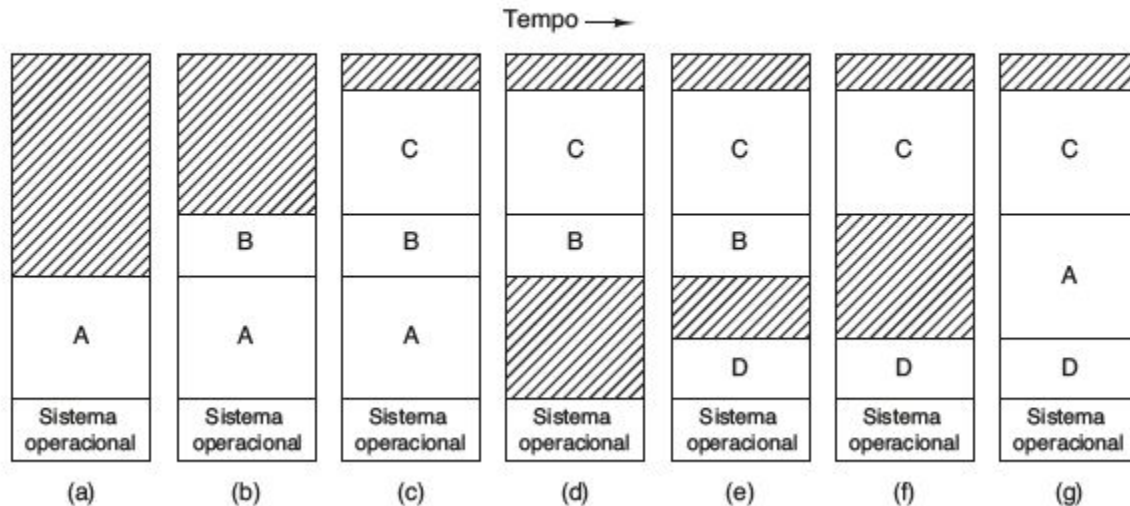
Swapping

- Principais estratégias:
- **Swapping**: consiste em trazer cada processo em sua totalidade, executá-lo por um tempo e então colocá-lo de volta no disco.
- **Memória Virtual**: permite que os programas possam ser executados mesmo quando estão apenas parcialmente na memória principal.

Swapping

- Os endereços precisam ser realocados sempre que os processos forem carregados da memória secundária.

FIGURA 3.4 Mudanças na alocação de memória à medida que processos entram nela e saem dela. As regiões sombreadas são regiões não utilizadas da memória.



Swapping

- Quanta memória deve ser alocada para cada processo?
- Essa quantidade é fixa (estática) ou pode mudar com tempo (dinâmica)?
 - Estática: o SO aloca o que é necessário.
 - Dinâmica: o processo pode necessitar de mais memória durante a execução (alocação dinâmica de memória).
 - Ele pode crescer se a posição adjacente estiver livre.
 - Ele pode ser movido ou trocado com outros para que a posição adjacente esteja livre.
 - Se nada for possível, ele será suspenso ou terminado.

Swapping

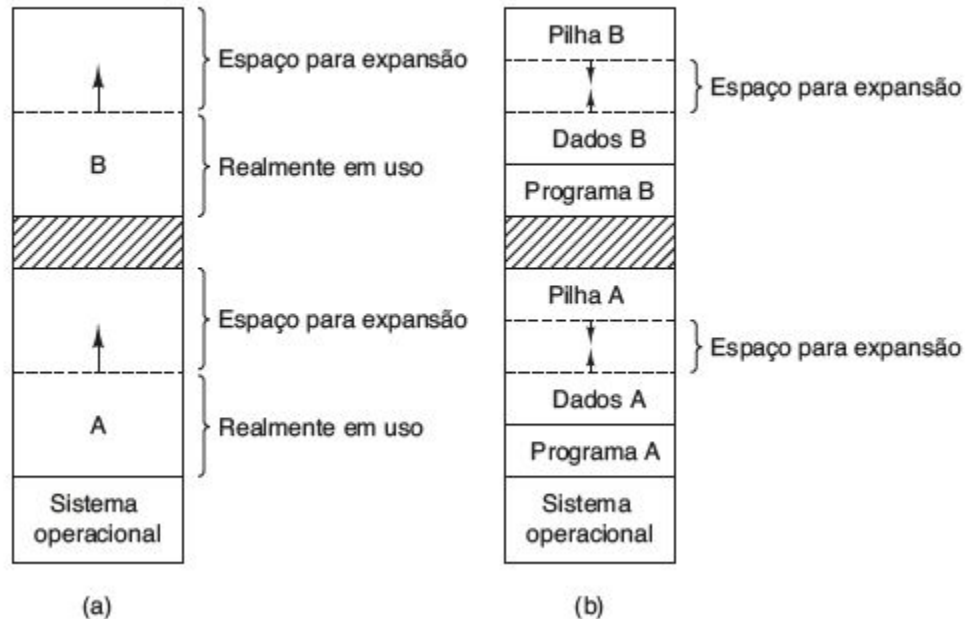
- Também podemos alocar um pouco de memória extra, se é esperado que o processo cresça.
 - A memória extra é alocada sempre que o processo é movido ou trocado de posição na memória.
 - No disco, armazenamos somente a porção em uso.
 - Desperdício de espaço e processamento ao armazenar o espaço extra.

Swapping

- Geralmente, os processo apresentam dois segmentos de expansão:
 - Área de pilha: variáveis locais normais e endereços de retorno.
 - Segmentos de dados: usados como uma área temporária para variáveis que são dinamicamente alocadas e liberadas.

Swapping

FIGURA 3.5 (a) Alocação de espaço para um segmento de dados em expansão. (b) Alocação de espaço para uma pilha e um segmento de dados em expansão.



Swapping

- Na parte b) da imagem, observe que o espaço extra pode ser usado por qualquer segmento.
- Se ela acaba, o processo poderá:
 - Ser transferido para outra área com espaço suficiente;
 - Ser transferido para o disco até que um espaço de tamanho suficiente possa ser criado;
 - Ser terminado.

Memória Virtual

Memória Virtual

- O tamanho dos softwares estão crescendo mais rápido que o tamanho da memória RAM.
 - Jogos digitais modernos AAA: 8GB, 16GB 32GB de RAM.
 - Diablo 4: recomendado 16GB de RAM
 - Windows 11: recomendado 8GB de RAM
- Alguns programas são grandes demais para a memória principal!
 - Imagine o cenário com múltiplos programas.

Memória Virtual

- O swapping não é recomendado nessa situação.
 - Tempo de transferência.
 - Espaço disponível na memória RAM.
- Solução primitiva: separar o código em módulos pequenos, chamados de sobreposições.
 - Primeira ação do programa consistia em carregar o gerenciador de sobreposições.
 - Quando uma sobreposição encerrava, outra era carregada em uma posição livre ou sobreposta em outra.
 - Gerenciado pelo programados.

Memória Virtual

- Solução por memória virtual: cada programa tem seu próprio espaço de endereçamento, o qual é dividido em blocos chamados de páginas.
 - Cada página é uma série contígua de endereços.
 - Elas são mapeadas na memória física, mas nem todas precisam estar na memória física ao mesmo tempo para executar o programa.

Memória Virtual

- Quando o programa referencia uma parte do espaço de endereçamento que está na memória física, o hardware realiza o mapeamento necessário rapidamente.
- Quando o programa referencia uma parte de seu espaço de endereçamento que não está na memória física, o sistema operacional é alertado para ir buscar a parte que falta e reexecuta a instrução que falhou.
 - Enquanto um programa está esperando que partes de si mesmo sejam lidas, a CPU pode ser dada para outro processo.

Resumo

Resumo

- Espaço de Endereçamento
 - Proteção e Realocação
 - Realocação Dinâmica
- Swapping
- Memória Virtual

Dúvidas?

