

Sistemas Operacionais

Gerência de Memória - Introdução

Gerência de Memória

Todo programador deseja uma memória infinitamente grande, rápida e não volátil.

Infelizmente a tecnologia atual não oferece diretamente tais exigências.

Atualmente a maioria dos computadores possui uma hierarquia de memória para tentar sanar o problema.

Gerência de Memória

Hierarquia de memória:

- Uma pequena memória cache, muito rápida e de alto custo.
- Uma média memória principal (RAM), atualmente com alguns gigabytes. Velocidade média e médio custo.
- E uma grande memória secundária (disco), atualmente com centenas de gigabytes. Baixa velocidade e baixo custo.

Gerência de Memória

É função do Gerenciador de Memória controlar quais partes estão em uso, e quais não estão.

- Alocar memória para novos processos.
- Espaço de endereçamento.
- Liberar memória dos processos que terminam.
- Alocar mais memória para processos que demandam.
- Efetuar a troca de processos (swapping), quando a RAM não suporta todos os processos ao mesmo tempo.

Gerência de Memória

Existem duas classes para o gerenciamento de memória:

- Sistemas que efetuam o swapping e;
- Sistemas que não efetuam o swapping.

Se um dia existir tecnologia que disponibiliza mais memória que os programas podem ocupar, poderemos seguir para os algoritmos mais simples.

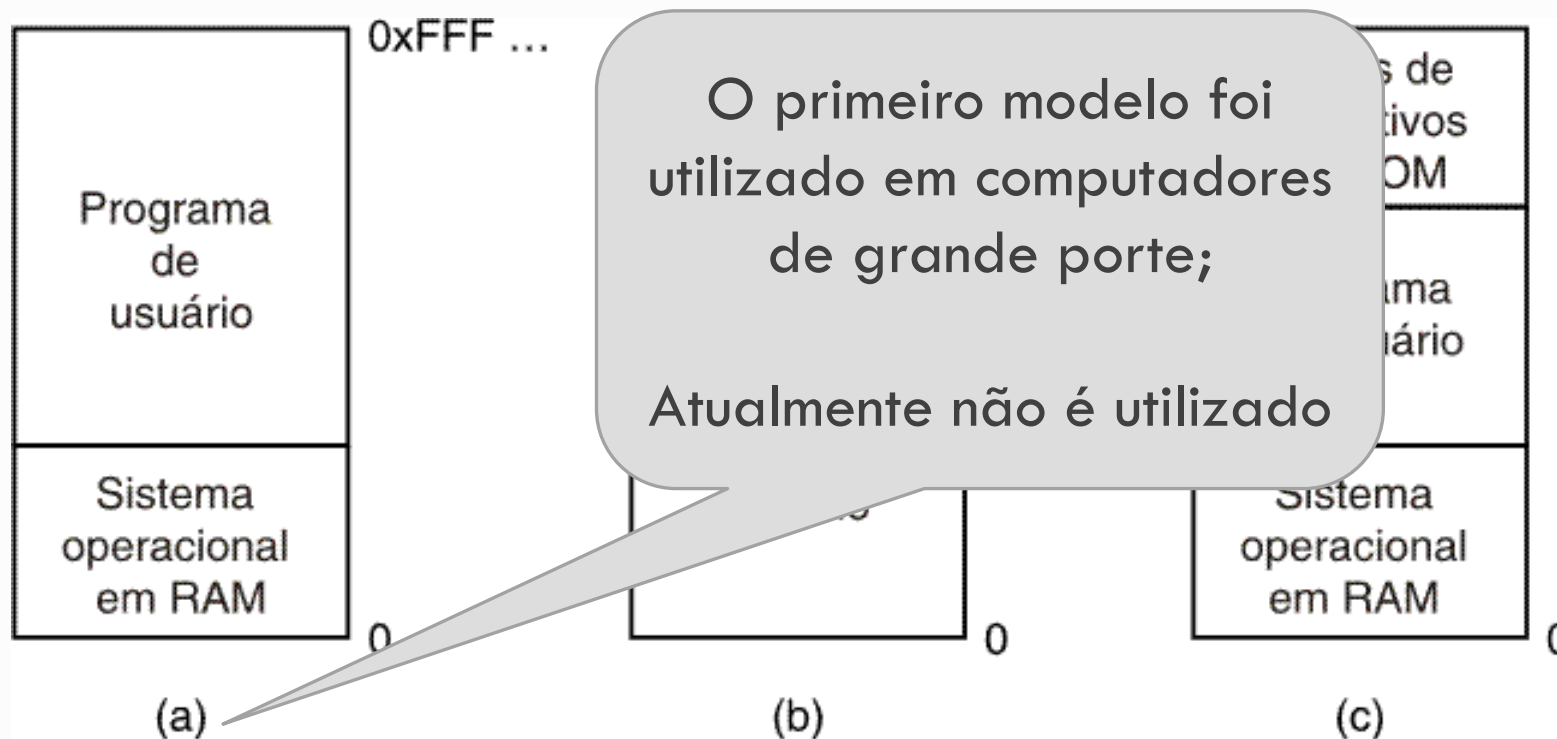
Monoprogramação sem troca de processos

No máximo um processo reside na memória em um mesmo instante de tempo.



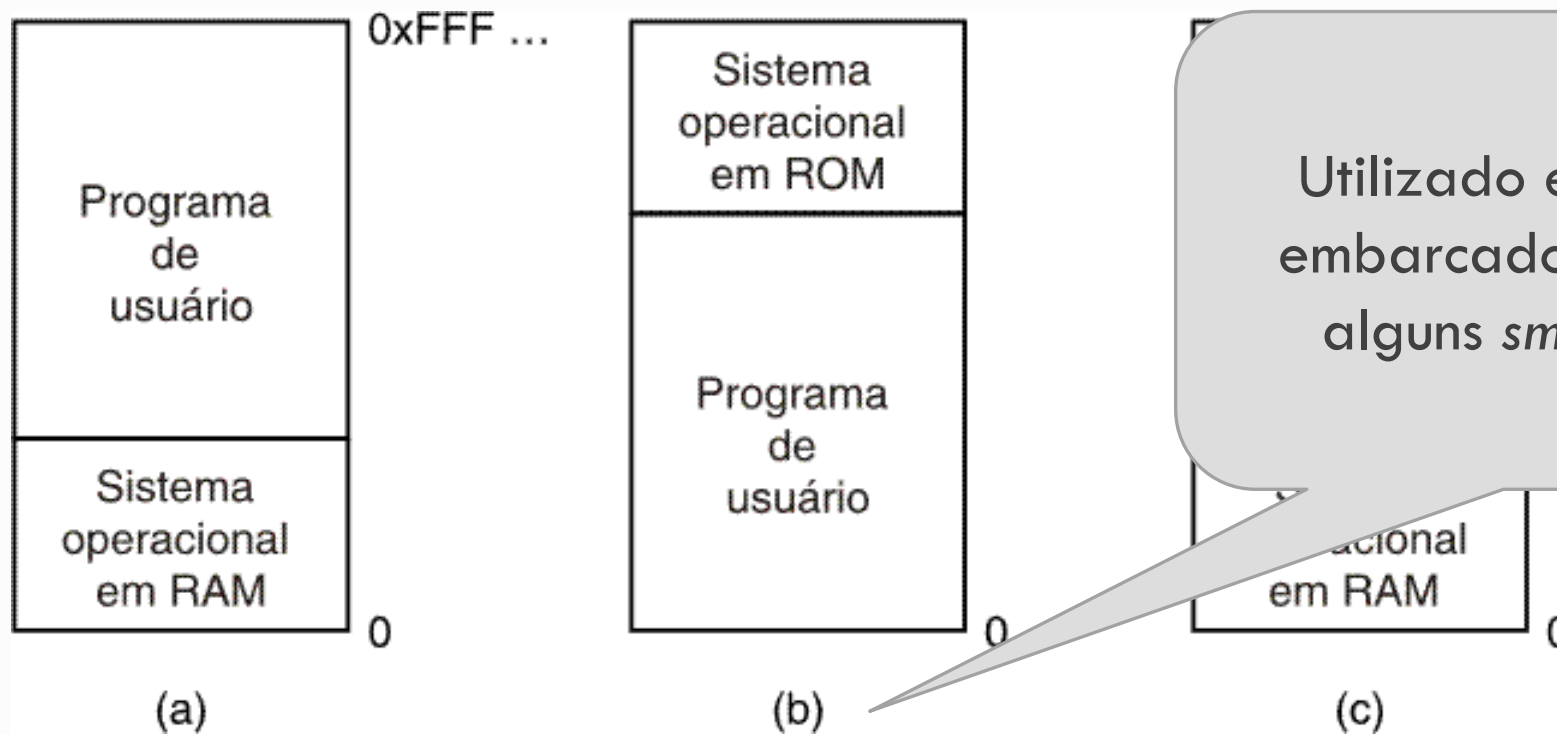
Monoprogramação sem troca de processos

No máximo um processo reside na memória em um mesmo instante de tempo.



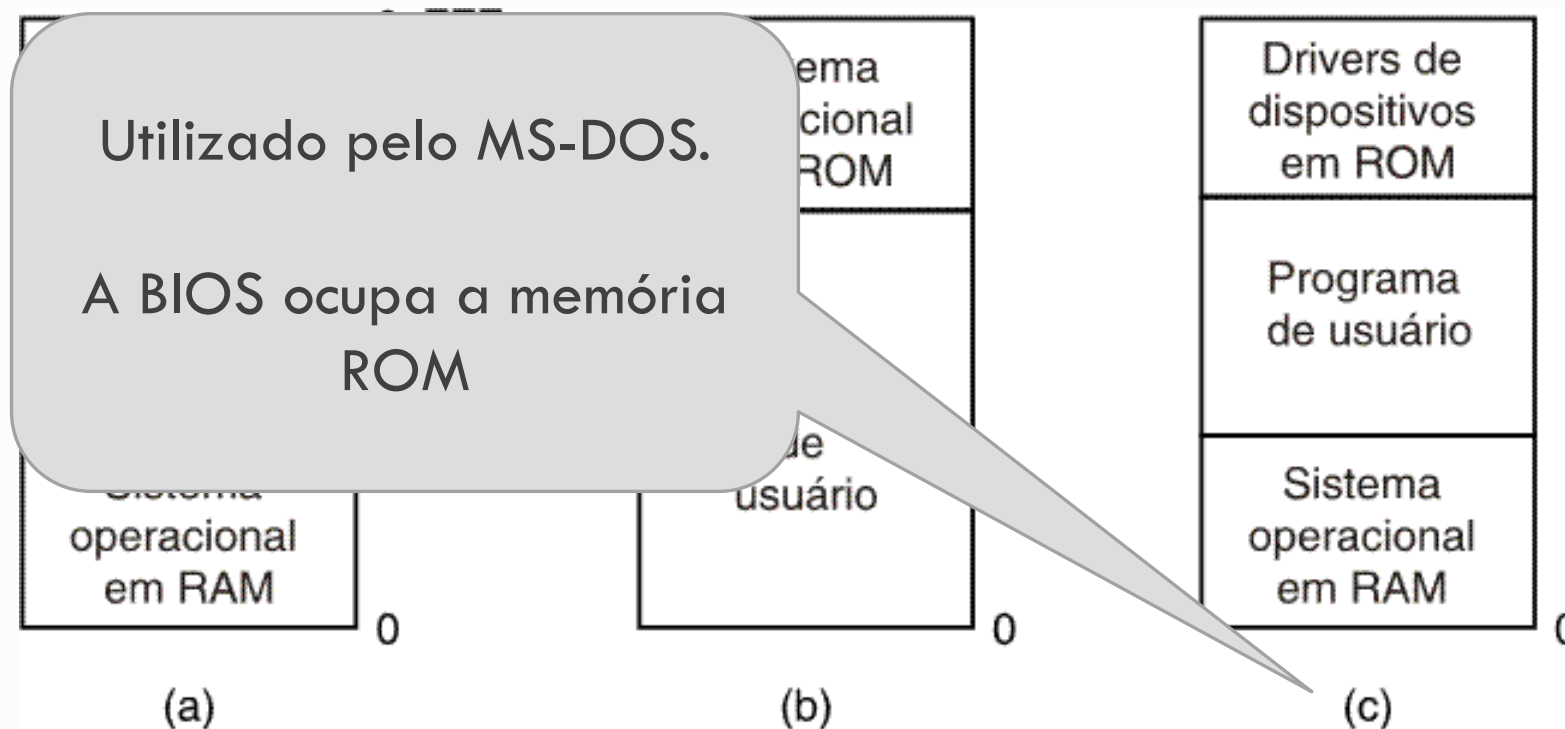
Monoprogramação sem troca de processos

No máximo um processo reside na memória em um mesmo instante de tempo.



Monoprogramação sem troca de processos

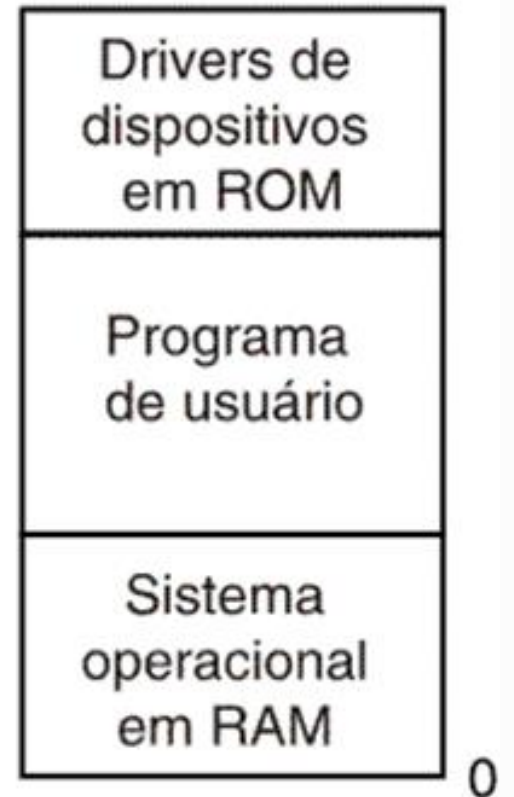
No máximo um processo reside na memória em um mesmo instante de tempo.



Monoprogramação sem troca de processos

Quando o programa finaliza, o espaço reservado para o programa de usuário é liberado e o sistema operacional indica no *prompt* o termino do processo.

Neste instante ele aguarda uma nova entrada no terminal para carregar um novo processo na memória.

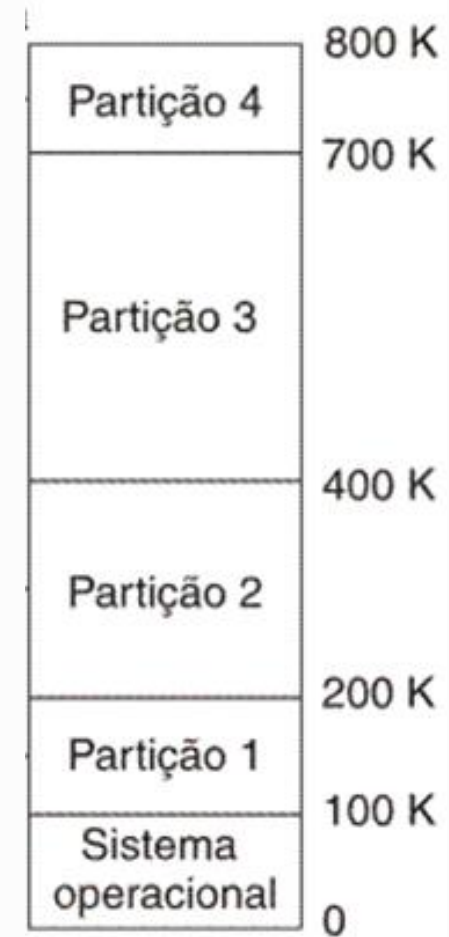


(c)

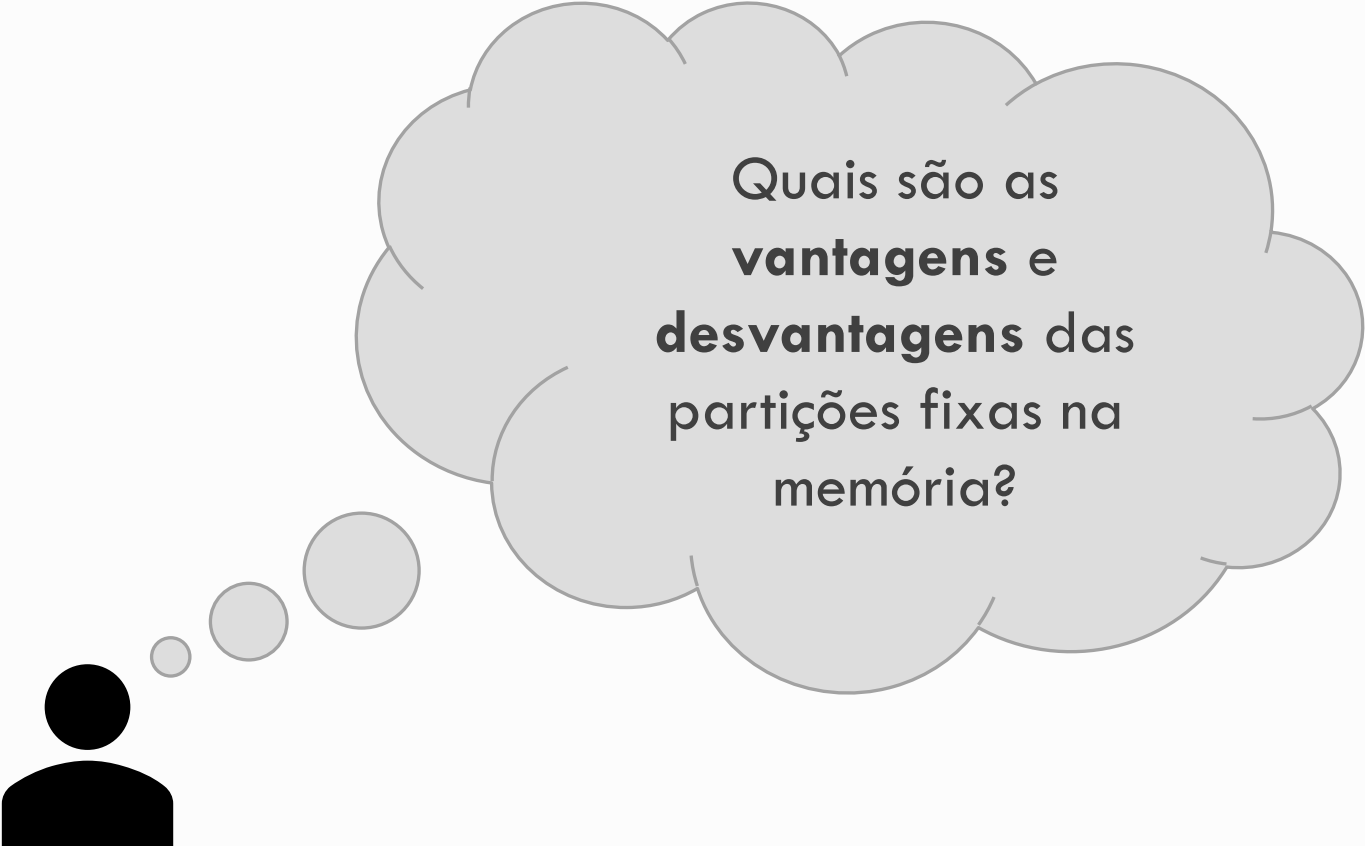
Multiprogramação com partições fixas

Neste modelo vários processos são aceitos na memória ao mesmo tempo (necessário para a multiprogramação).

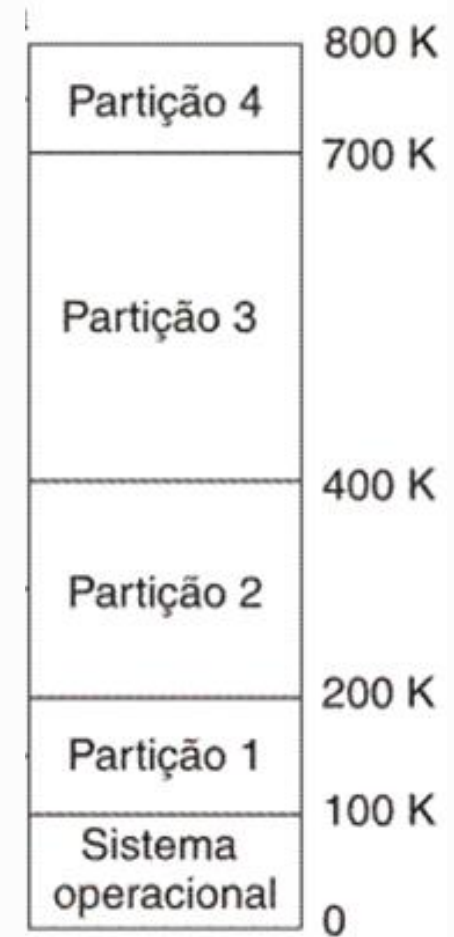
A maneira mais simples de suportar a multiprogramação em memória é dividindo a memória principal em partições fixas.



Multiprogramação com partições fixas



Quais são as **vantagens e desvantagens** das partições fixas na memória?



Multiprogramação com partições fixas

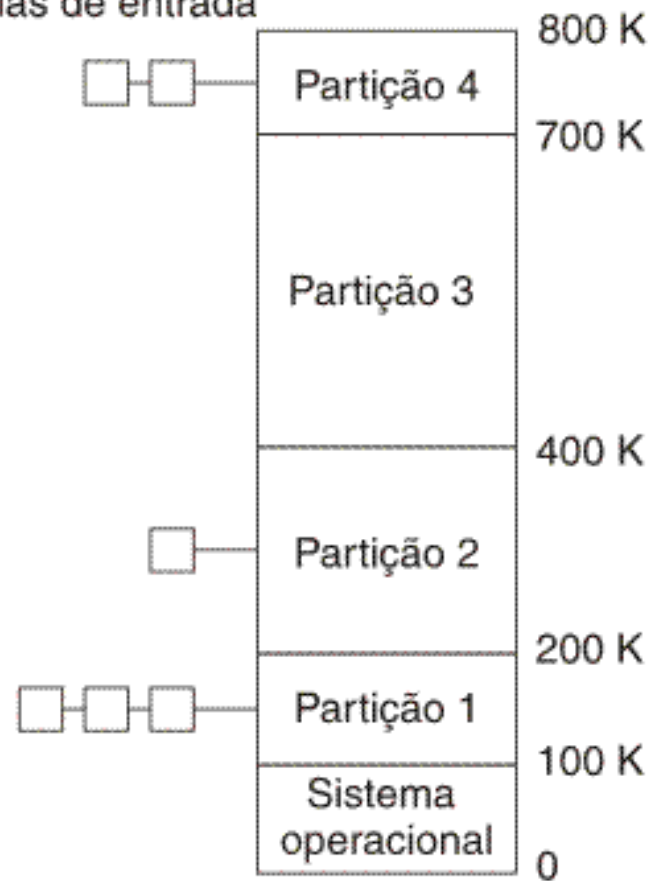
Sempre que um novo processo é carregado, ele é inserido em uma lista de processos que aguardam um espaço na memória para serem executados.

Basicamente, dois modelos podem ser implementados:

- Utilizar uma lista para cada partição de memória.
- Utilizar uma lista geral, para todas as partições.

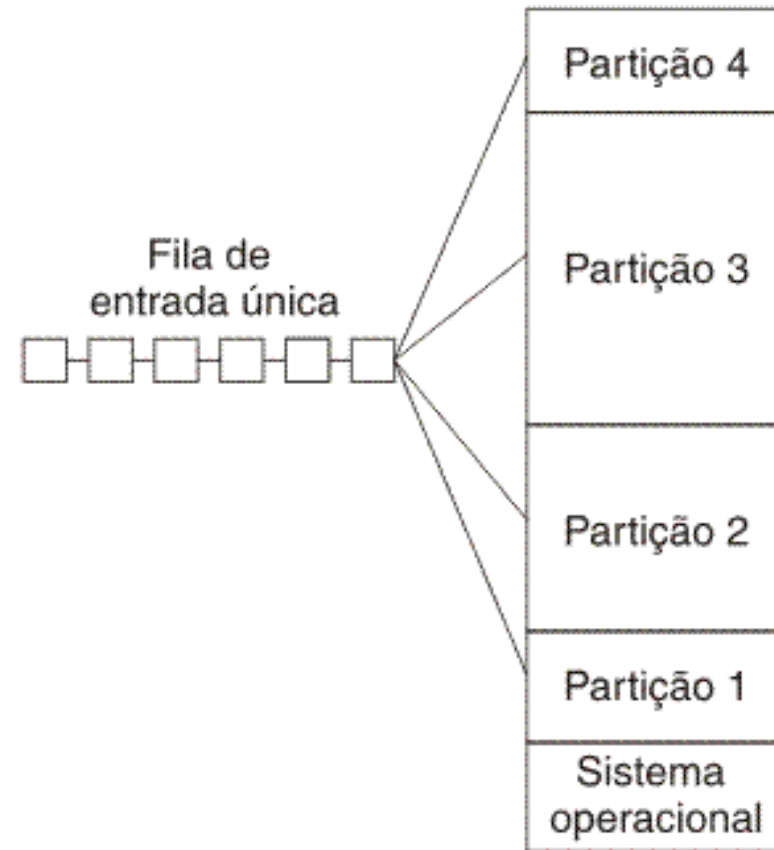
Multiprogramação com partições fixas

Múltiplas filas de entrada



(a)

Fila de entrada única



(b)

Troca de Processos

Existem duas formas de gerenciar processos em memória:

- Troca de processos (swapping).

Todo processo precisa estar na memória principal para ser executado.

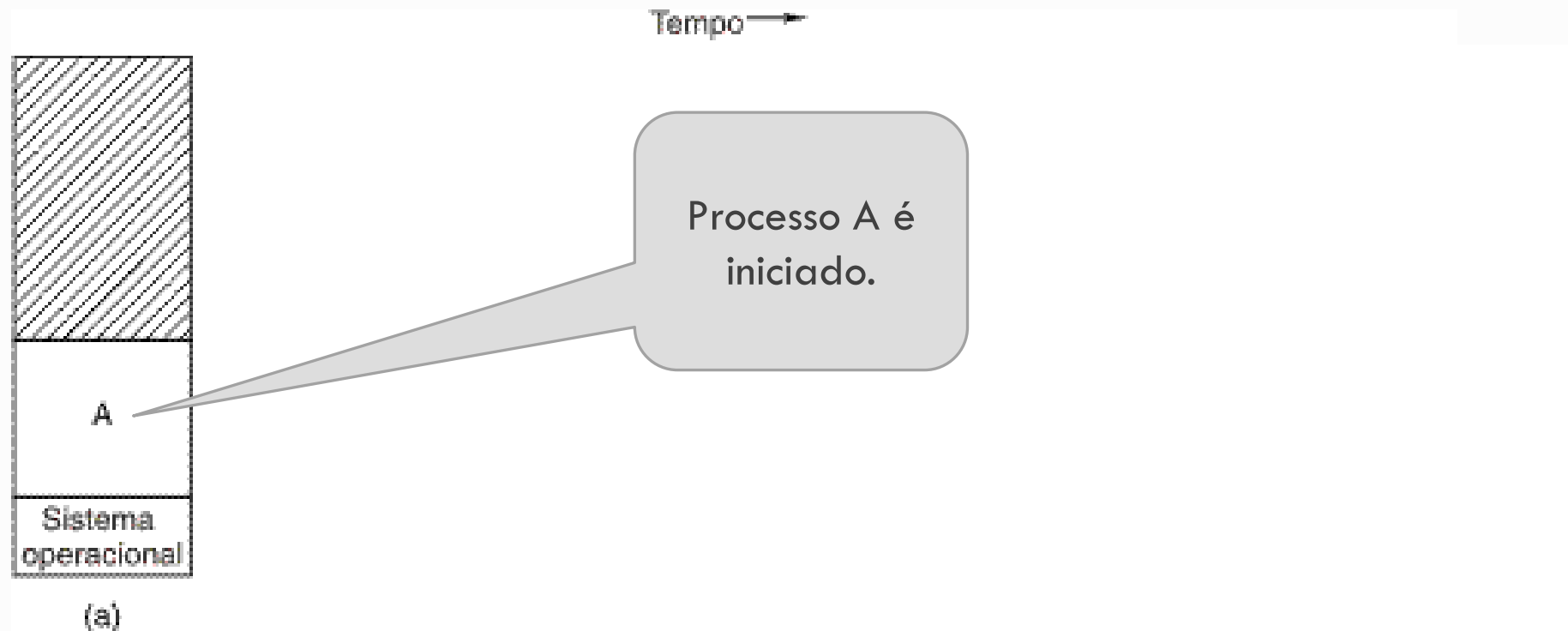
- Memória virtual

Permite que programas possam ser executados mesmo que estejam parcialmente na memória principal (será detalhado nas próximas aulas).

Troca de Processos

Exemplo de alocação dinâmica de processos em partições de tamanhos variáveis:

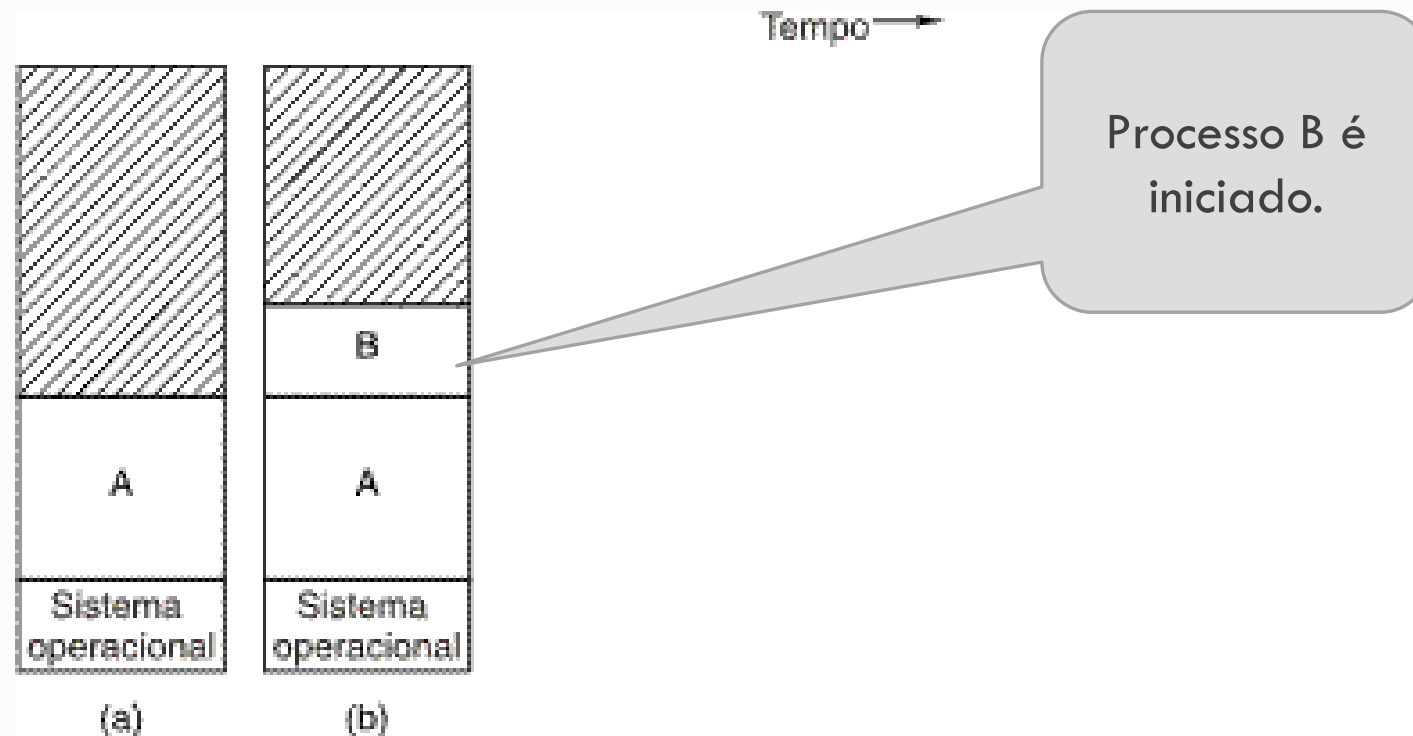
- Espaços sombreados são espaços livres na memória.



Troca de Processos

Exemplo de alocação dinâmica de processos em partições de tamanhos variáveis:

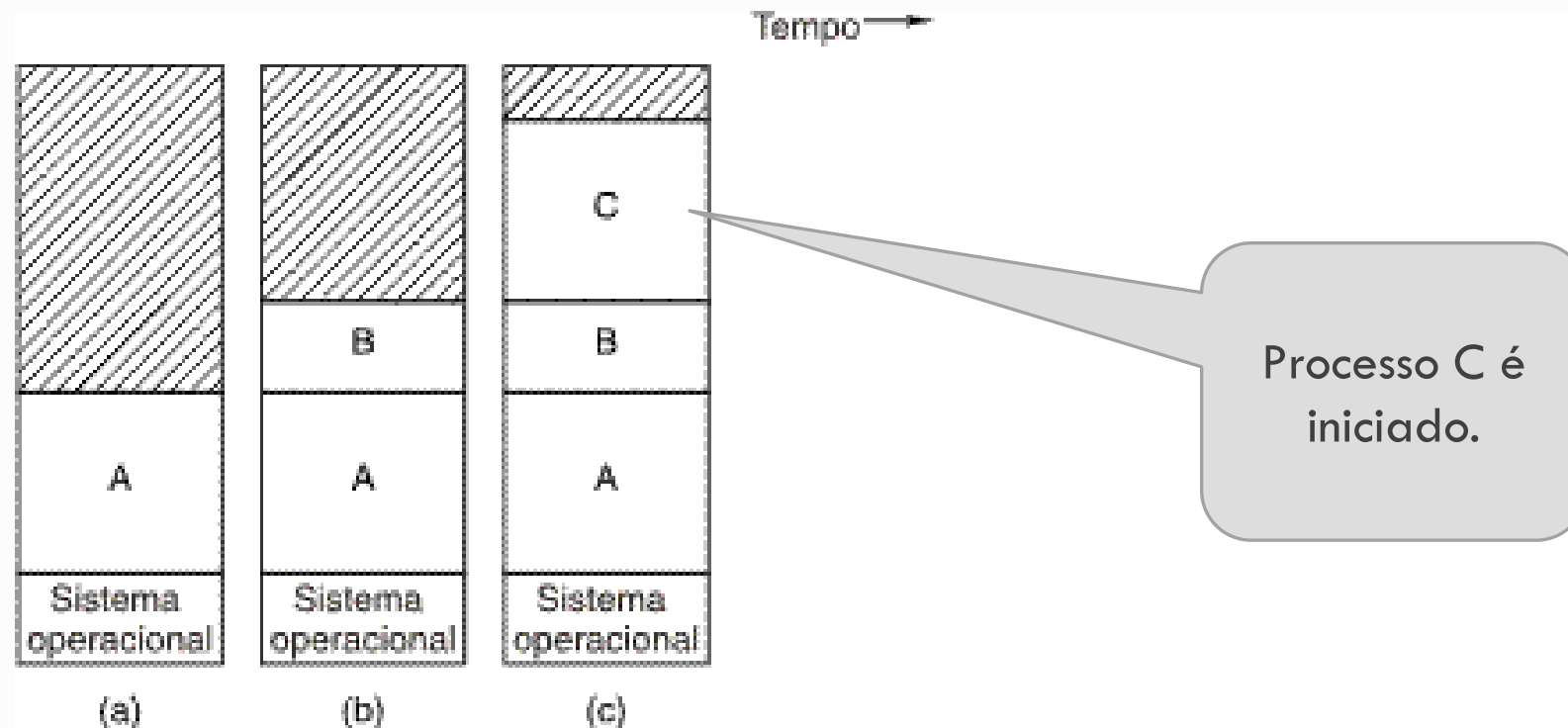
- Espaços sombreados são espaços livres na memória.



Troca de Processos

Exemplo de alocação dinâmica de processos em partições de tamanhos variáveis:

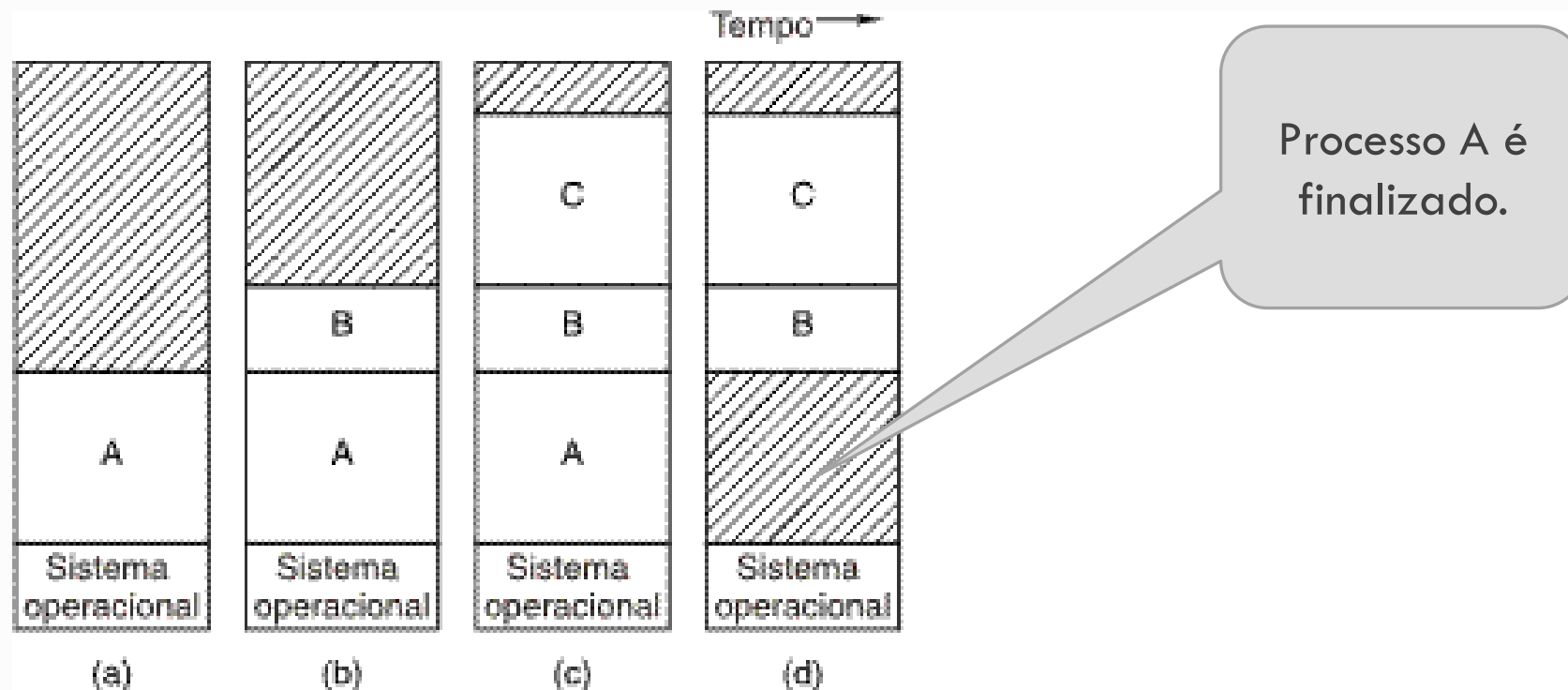
- Espaços sombreados são espaços livres na memória.



Troca de Processos

Exemplo de alocação dinâmica de processos em partições de tamanhos variáveis:

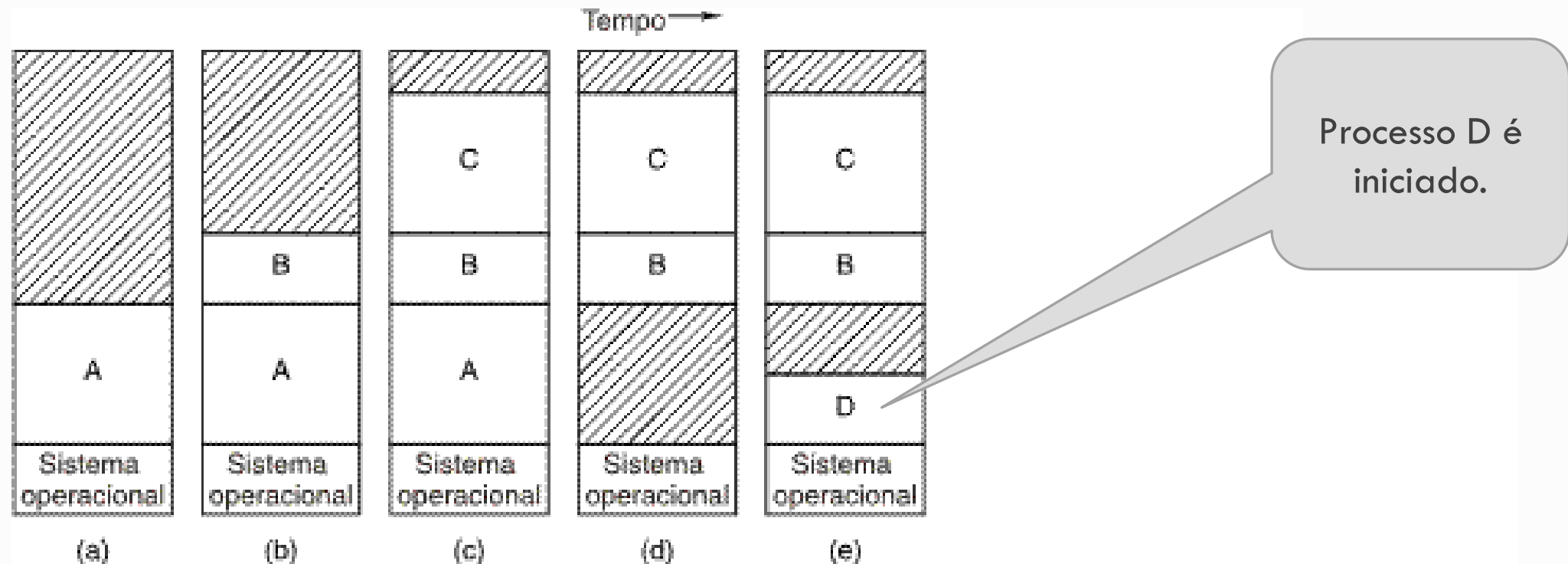
- Espaços sombreados são espaços livres na memória.



Troca de Processos

Exemplo de alocação dinâmica de processos em partições de tamanhos variáveis:

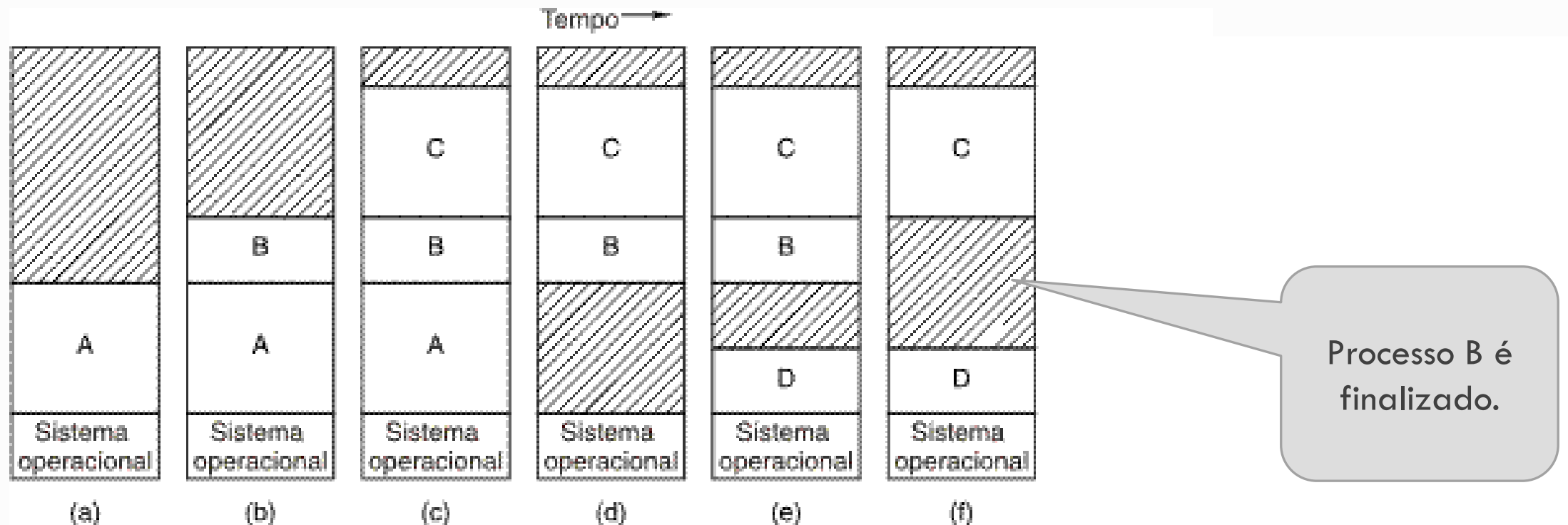
- Espaços sombreados são espaços livres na memória.



Troca de Processos

Exemplo de alocação dinâmica de processos em partições de tamanhos variáveis:

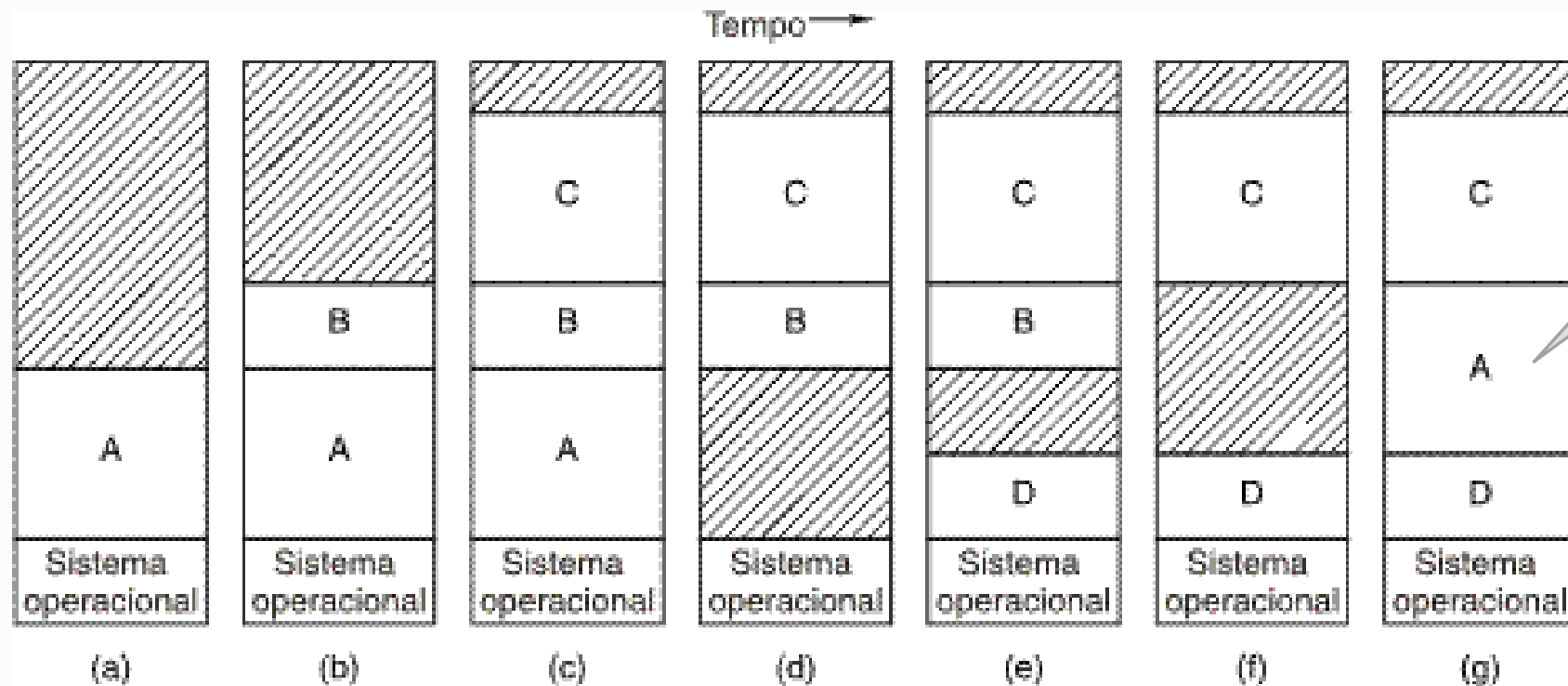
- Espaços sombreados são espaços livres na memória.



Troca de Processos

Exemplo de alocação dinâmica de processos em partições de tamanhos variáveis:

- Espaços sombreados são espaços livres na memória.

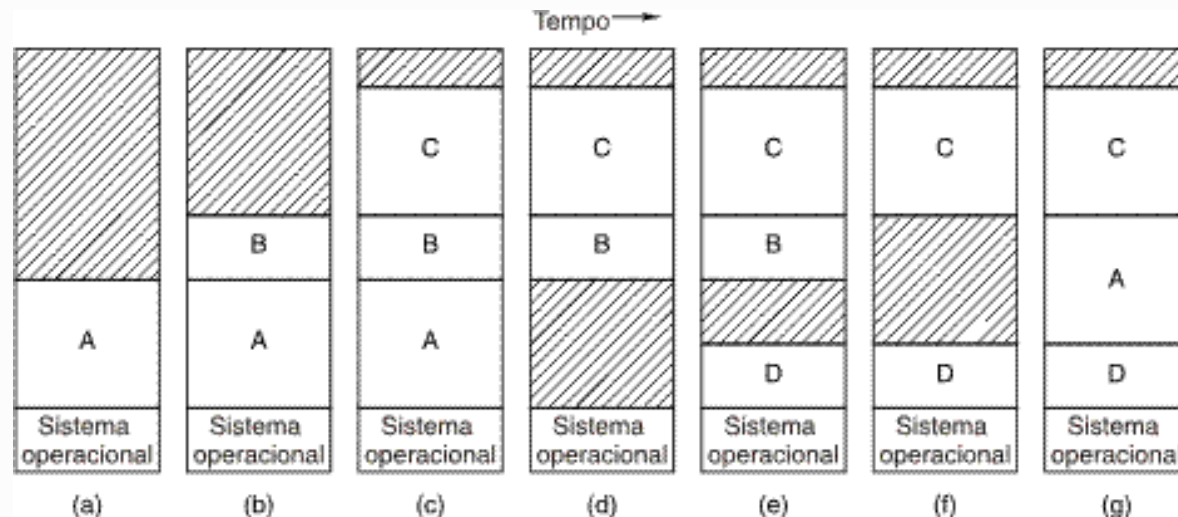


Processo A é iniciado.

Troca de Processos

A troca de processos pode deixar muitos espaços vazios na memória.

A solução é efetuar a compactação de memória. Geralmente não é uma prática comum nos S.O.s por gastar muito tempo para realocar todos os processos.



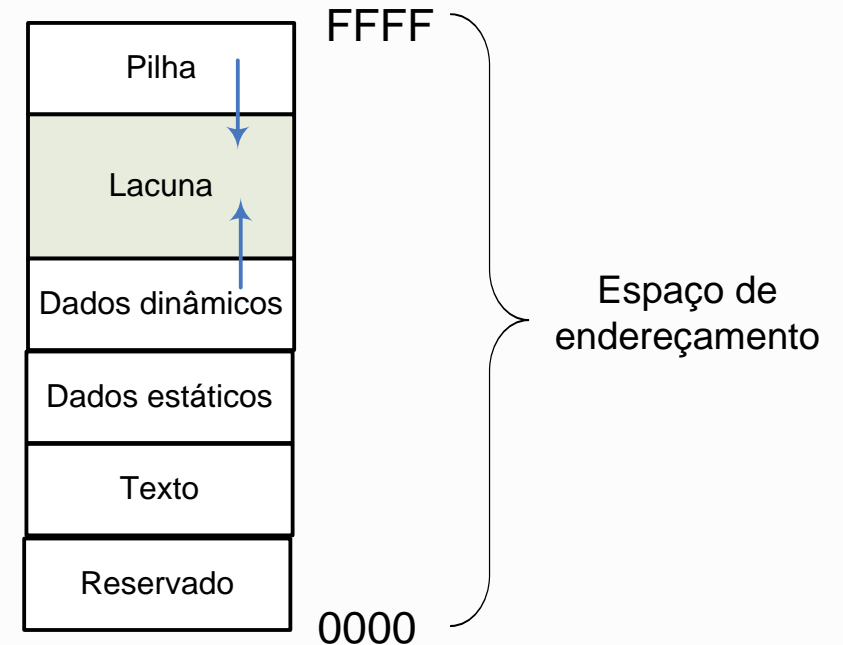
Troca de Processos

Um ponto importante é a quantidade de memória que deve ser alocada para cada processo.

- Se o tamanho é fixo, a implementação é simples.

Contudo, lembre-se do espaço de endereçamento:

- Atualmente, para os S.O.s comerciais, o tamanho dos processos é fixo?



Troca de Processos

Na necessidade de mais memória por processo, se houver espaço disponível adjacente suficiente ao processo, o algoritmo é simples.

Podemos simplesmente alocar o espaço disponível ao processo em crescimento.

Se não houver espaço livre adjacente, ou espaço livre adjacente suficiente, o processo deve ser realocado na memória, para uma região mais promissora.

Mas este espaço para realocação pode não existir...

Troca de Processos

Se não existir uma partição onde o processo em crescimento caiba, o mesmo deve:

- Aguardar na memória secundária, ou;
- O S.O. deve descartá-lo (*kill idProcess*);

Por quê não deve aguardar na memória primária?

Troca de Processos

Obviamente, os S.O.s modernos alocam uma memória extra para os processos, devido:

- A alocação dinâmica de variáveis.
- Ao o crescimento da pilha de execução.

A JVM, por exemplo, suporta parâmetros no momento da inicialização de um processo Java onde o usuário pode informar:

- Tamanho inicial do processo.
- Tamanho máximo do processo.

Troca de Processos

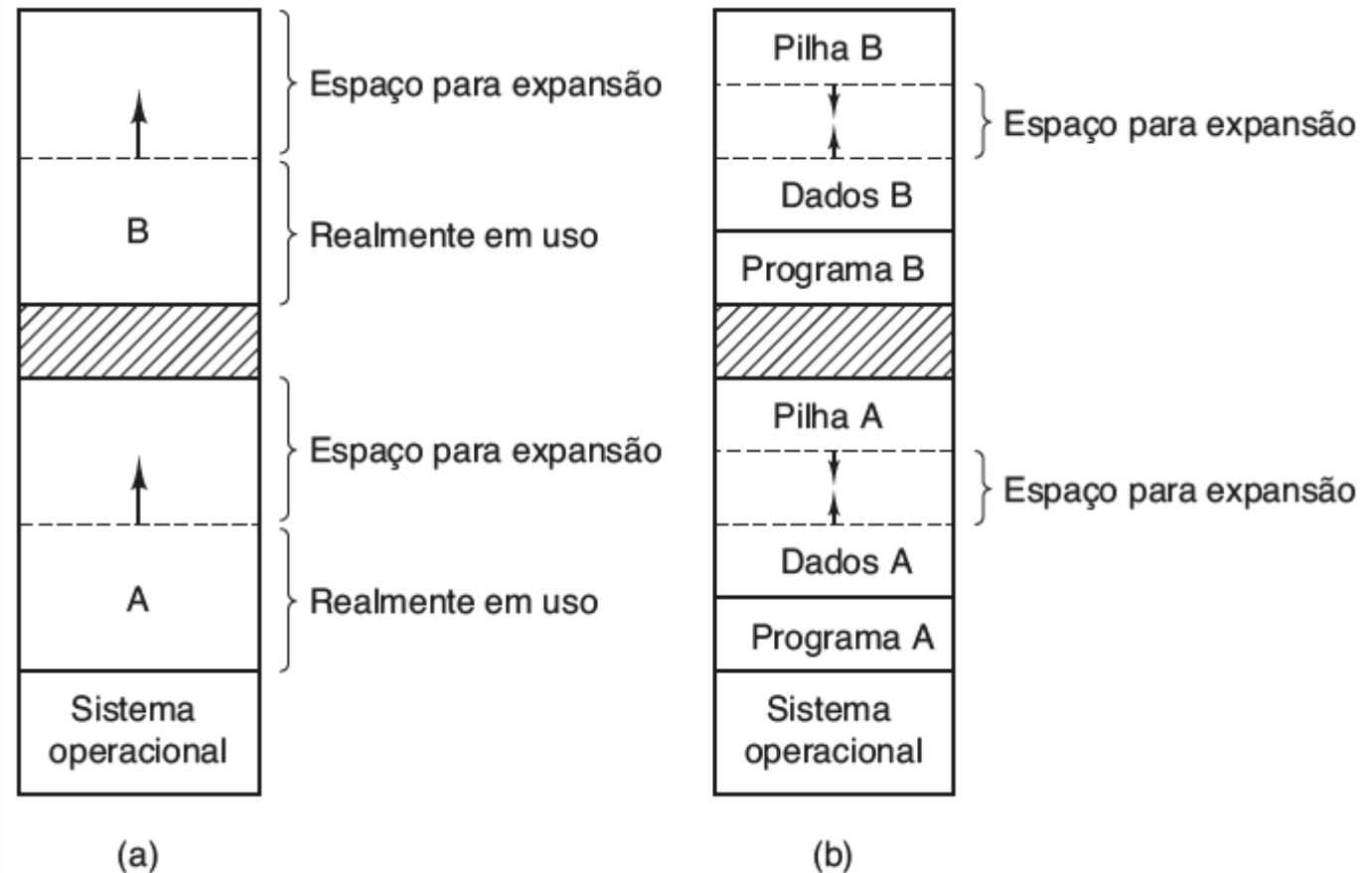
Observação (relacionado a eficiência do sistema):

- Quando o processo for transferido para o disco, somente a memória em uso deve ser transportada, economizando na quantidade de operações necessárias.

Troca de Processos

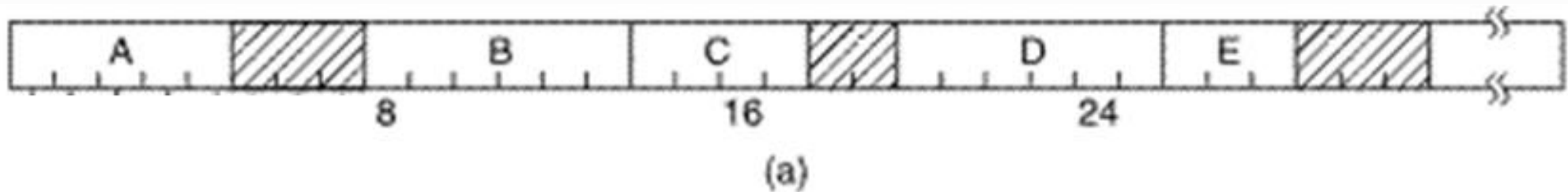
a) Crescimento apenas da *heap*;

b) Crescimento da *heap* e da pilha de execução.

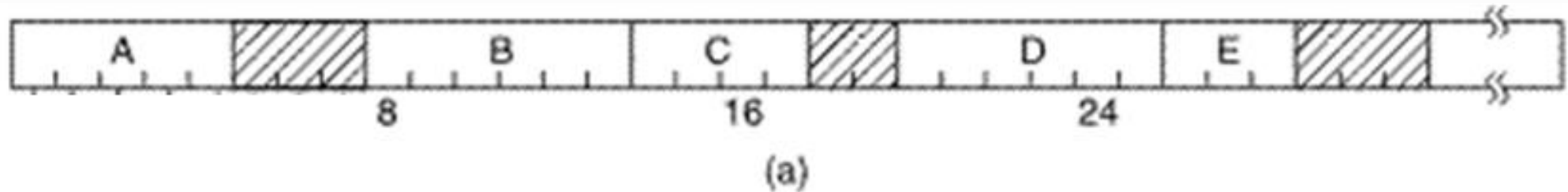


Representação

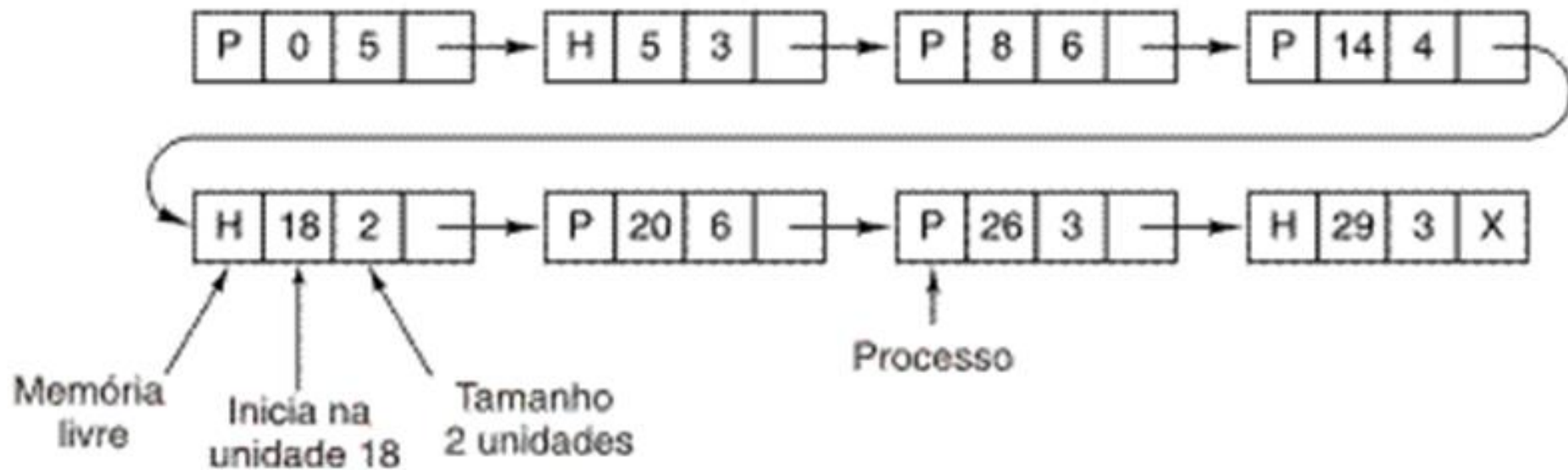
Suponha a memória abaixo, suportando os processo A, B, C, D, e E:



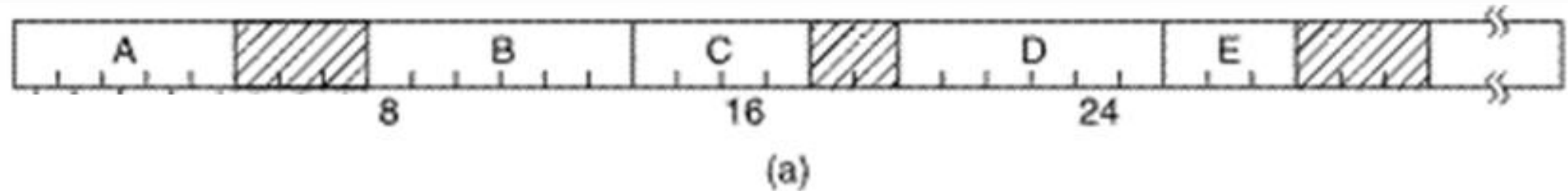
Representação



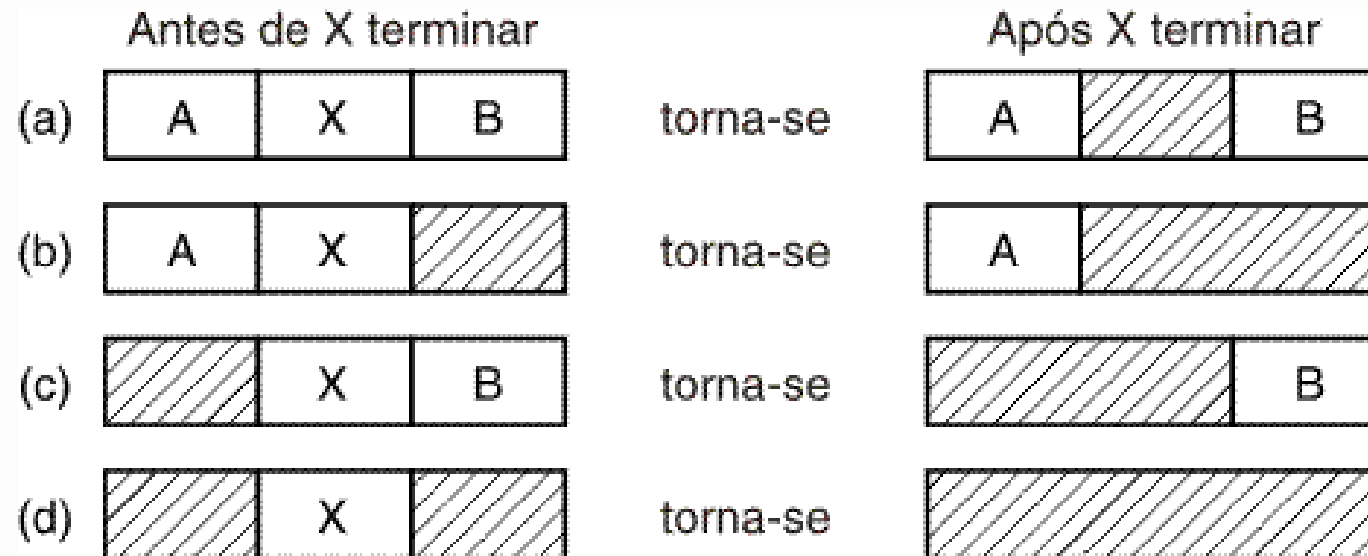
Uma possível implementação é por lista encadeada.

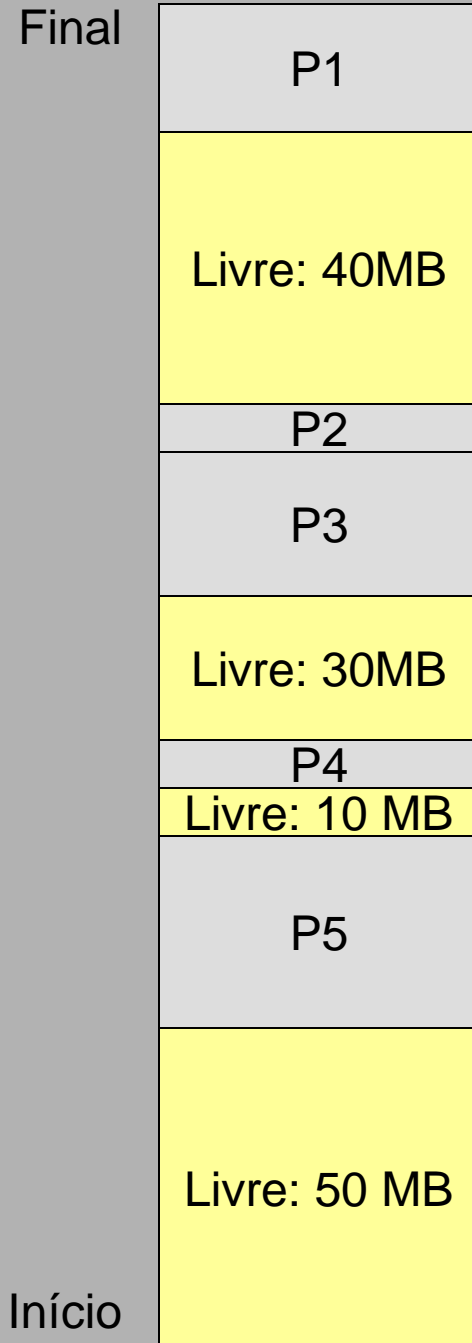


Representação



Existem quatro possibilidades quando um processo termina sua execução:

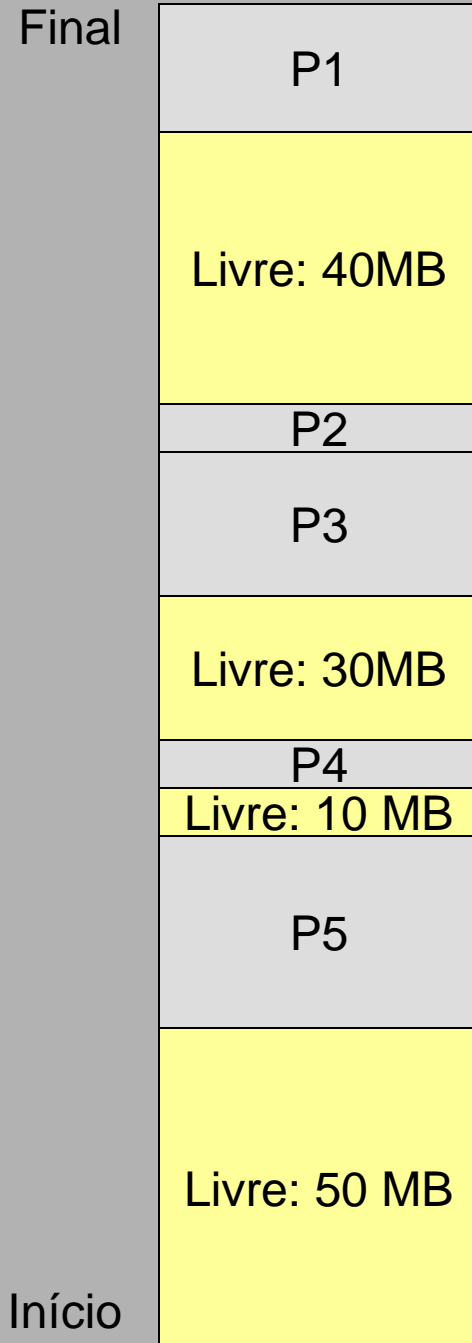




Algoritmos para Gerência de Memória

First-Fit: Procura primeiro segmento que couber o processo;

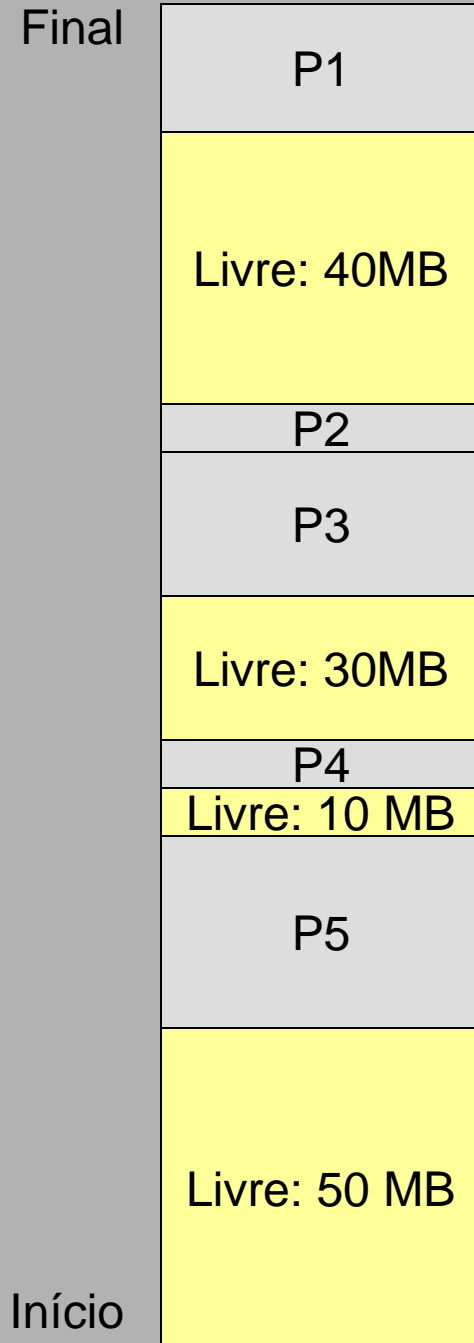
- Suponha o processo de tamanho 10MB;
- Onde ele será alocado?



Algoritmos para Gerência de Memória

Next-Fit: com memorização resultado da última alocação (parecido com o first-fit).

- Suponha o processo de tamanho 35MB;
- O último processo alocado foi o P4;
- Onde ele será alocado?

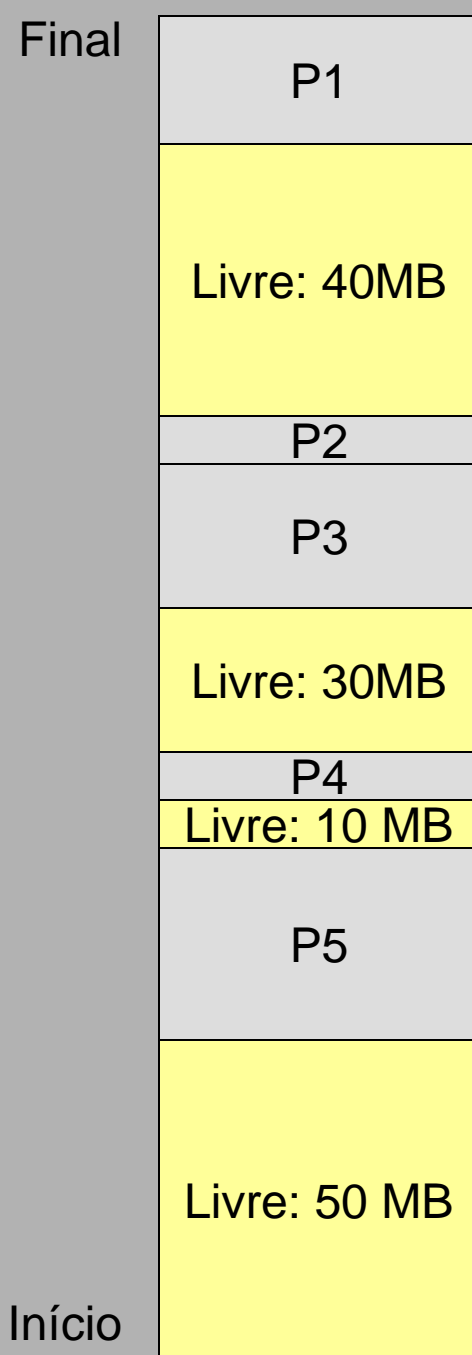


Algoritmos para Gerência de Memória

Best-Fit: Procura o menor segmento que caiba o Processo.

Oferece uma sobra mínima dentre as possibilidades.

- Suponha o processo de tamanho 28MB.
- Onde ele será alocado?



Algoritmos para Gerência de Memória

Worst-Fit: Procura o maior segmento disponível, o restante ainda deve servir.

Oferece uma sobra máxima dentre as possibilidades:

- Suponha o processo de tamanho 3MB;
- Onde ele será alocado?

Algoritmos para Gerência de Memória

O problema relacionado a fragmentação existente no Gerenciamento de Sistemas de Arquivos também existe na gerência de memória.

Experimentos computacionais mostram que não existe uma estratégia que DEVE ser utilizada na alocação de processos.

- Mostram apenas que as políticas mais simples funcionam muito bem (como o next-fit).

Próxima aula

Leitura:

Sistemas operacionais modernos

- Memória Virtual e Paginação

Referências

Sistemas Operacionais Modernos. Tanenbaum, A. S. 2ª edição. 2003.

Sistemas Operacionais. Conceitos e Aplicações. A. Silberschatz; P. Galvin; G. Gagne. 2000.

Sistemas Operacionais – Projeto e Implementação. Tanenbaum, A. S. 2ª edição. 2000.

Slides Prof. Humberto Brandão