4.2 - Gerenciamento de Memória com Swapping

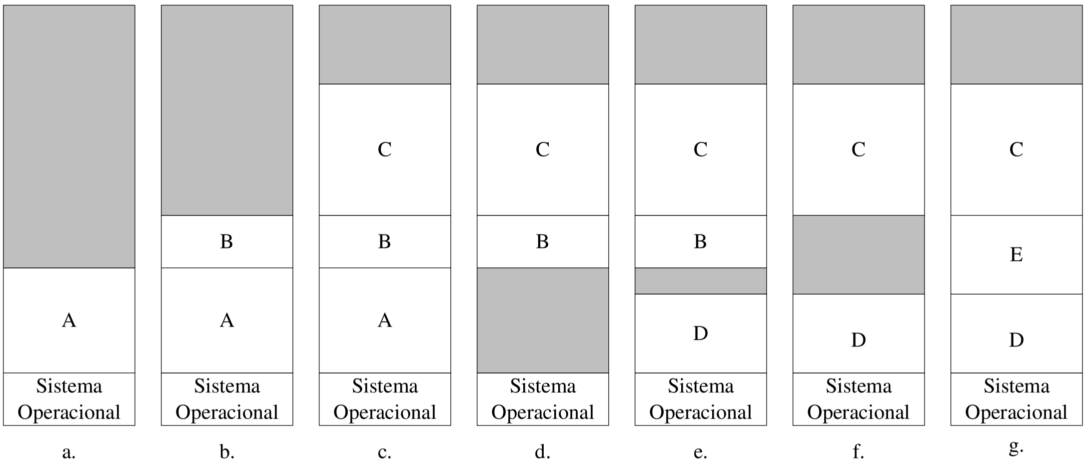
- ... como discutido anteriormente, em um sistema multiprogramado em lote, a organização em partições fixas é simples e efetiva — quanto mais *jobs* na memória, mais a CPU estará ocupada e não há razão para usar algo mais complicado;
- nos sistemas *time-sharing*, a situação é diferente uma vez que há mais usuários utilizando o sistema que memória disponível para os seus processos;
- ... neste contexto, mover temporariamente processos para o disco é a alternativa para atender os usuários em suas requisições;
- ... este processo de permuta entre a memória principal e o disco, como o intuito de criar espaço na memória para executar outros processos é denominado de *swapping*.

4.2.1 - Multiprogramação com Partições Variáveis

- em princípio, um sistema que utiliza *swapping* pode ser baseado em partições fixas, assim, sempre que um processo é bloqueado, haverá um permuta entre este e um outro processo que esteja no disco;
- ... na prática, partições fixas são pouco atrativas quando a área da memória é escassa fazendo com que muita memória seja perdida por programas cujo tamanho é bem inferior ao tamanho das partições;
- ... assim sendo, um novo sistema de gerenciamento de memória é proposto, o gerenciamento de memória com partições de tamanhos variáveis;
- **multiprogramação com partições variáveis** - o número, a localização e o tamanho das partições variam dinamicamente ao longo do tempo;
- ... a flexibilidade de não se ter um número fixo de partições aumenta a utilização da memória, mas também complica a tarefa de alocar, liberar e gerenciar a memória.

... 4.2.1 - Multiprogramação com Partições Variáveis

- ... quando partições variáveis são usadas, o número e o tamanho dos processos na memória varia dinamicamente como pode ser visto no diagrama abaixo:



- ... a principal diferença entre partições fixas e variáveis é que o número, a localização e o tamanho das partições variam dinamicamente ao longo do tempo.

... 4.2.1 - Multiprogramação com Partições Variáveis

- É possível combinar todos os buracos disjuntos num único buraco movendo todos os processos para um lado da memória — esta técnica é conhecida como **compactação da memória**;

- ... esta técnica é pouco empregada pelo fato de requerer muito tempo de CPU, por exemplo, um microcomputador com 1 MB de memória e que pode copiar 1 byte por μs, gasta 1 s para compactar toda a memória.

- ... um ponto negativo neste método é saber o quanto de memória alocar para um processo, o que na verdade não é tão simples dado que os segmentos de dados e pilha de um processo tendem a crescer durante a sua execução;

- ... para exemplificar, considere os processos de alocação dinâmica de memória e a recursão presentes nos programas — ambos são típicos exemplos do crescimento dos segmentos de dados e segmento de pilha.

... 4.2.1 - Multiprogramação com Partições Variáveis

- 02 situações são possíveis quando da necessidade de expansão:

  ❶ se o processo necessitar expandir sua memória e existir um buraco adjacente, simplesmente o buraco pode vir a ser incorporado ao espaço de endereçamento do processo;

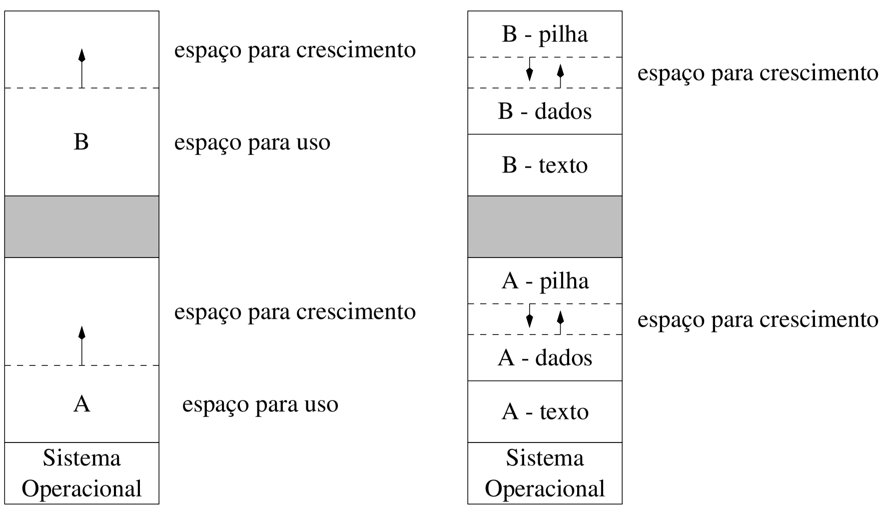
  ❷ de outra forma, se o processo está adjacente a outro processo, o primeiro deverá ser movido para um buraco grande o suficiente para armazená-lo, ou espaço de memória terá que ser criado com a remoção de algum processo.

- ... se for esperado que muitos processos cresçam na memória, uma boa política seria alocar uma pequena área extra toda vez que o processo é permutado ou movido;

- ... contudo, quando os processos são permutados para o disco, somente a área de memória atualmente em uso deve ser copiada, não havendo necessidade de efetuar cópia da área extra da memória reservada para o crescimento do processo.

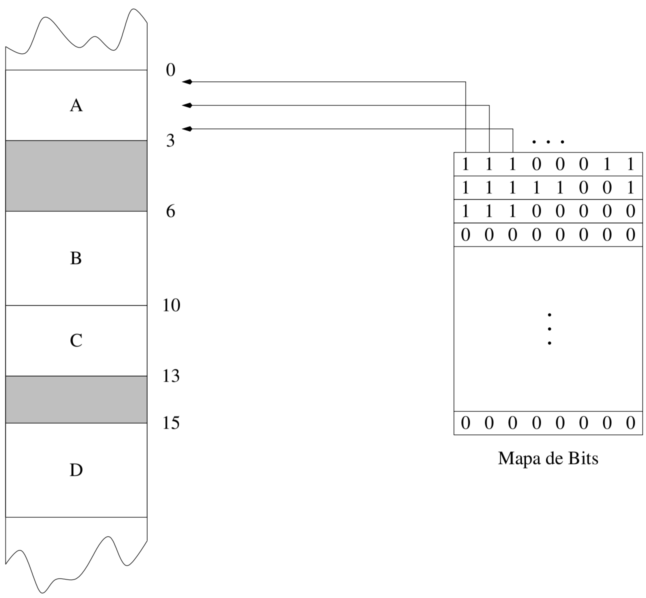
... 4.2.1 - Multiprogramação com Partições Variáveis

- ... a figura abaixo mostra a configuração da memória na qual a área para crescimento foi alocada para 02 (dois) processos:



4.2.2 - Gerenciamento de Memória com Mapa de Bits

- com um mapa de bits, cada unidade de alocação da memória tem um bit no mapa indicando se a unidade está livre ou sendo utilizada;



... 4.2.2 - Gerenciamento de Memória com Mapa de Bits

- ... o tamanho da cada unidade de alocação é uma importante fator de projeto — o que representa um gasto maior de memória no gerenciamento da própria memória;

- ... para pequenas unidades de alocação tem-se um mapa de bits maior, embora mesmo assim o montante gasto não necessariamente é grande, por exemplo, uma unidade de 32 bits irá requerer somente 3% da memória.

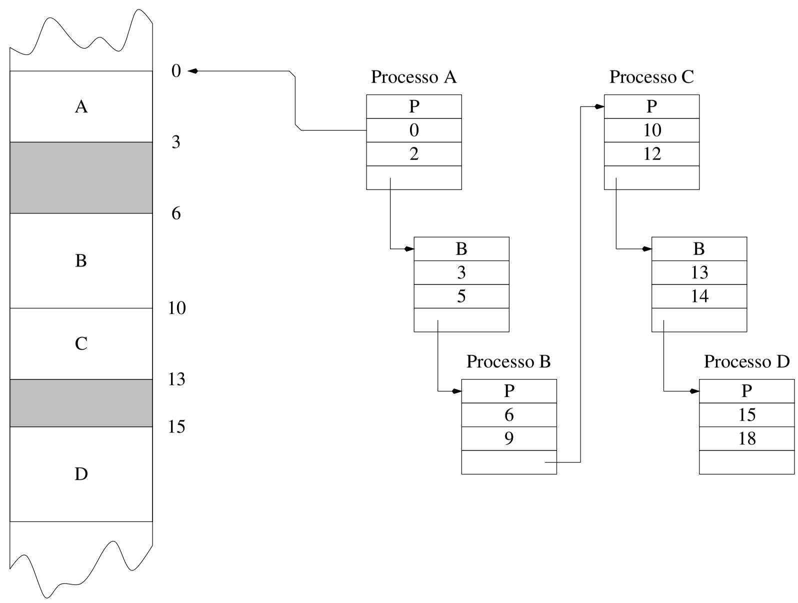
- ... se a unidade de alocação for escolhida grande, o mapa de bits será pequeno embora possa haver maior desperdício de memória caso o processo não seja um múltiplo exato da unidade de alocação;

✔ o mapa de bits depende somente do tamanho da memória e da unidade de alocação;

✘ para trazer um processo de k unidades de alocação para a memória, o gerenciador de memória deve pesquisar no mapa de bits uma seqüência de k consecutivos bits 0 no mapa — operação lenta, razão pela qual são evitados na prática.

4.2.3 - Gerenciamento de Memória com Listas Encadeadas

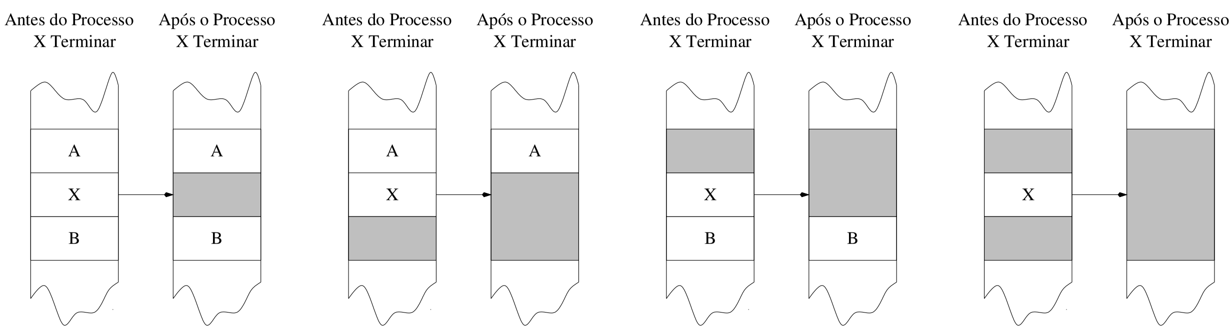
- outra maneira de gerenciar a memória é manter uma lista de alocações e segmentos de memória livre, onde um segmento é um processo ou um buraco entre 02 processos;



... 4.2.3 - Gerenciamento de Memória com Listas Encadeadas

- neste exemplo, a lista de segmentos é mantida ordenada por endereços que traz a vantagem de permitir a atualização direta da lista quando do término de um processo ou mudança do mesmo para o disco;

- ... quando do término de um processo, 04 situações podem ocorrer:

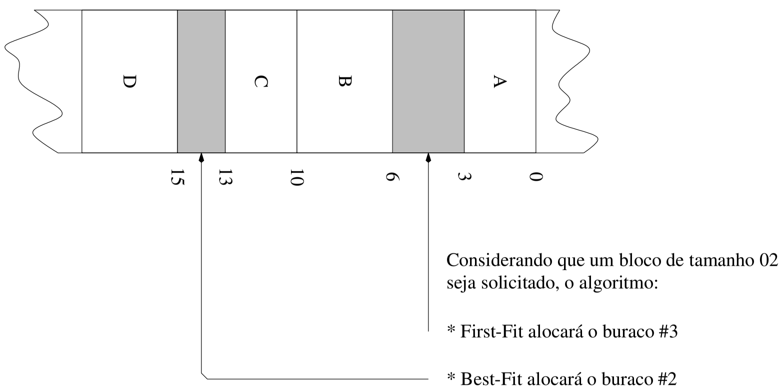


... 4.2.3 - Gerenciamento de Memória com Listas Encadeadas

- quando os processos e buracos são mantidos na lista ordenada por endereços, vários algoritmos podem ser usados para alocar memória:
- **algoritmo first-fit** - procura ao longo da lista de segmentos até encontrar um buraco de tamanho superior ou igual ao segmento de memória requisitado (M bytes);
- ... caso o buraco tenha tamanho superior (N bytes), o buraco é dividido em 02 segmentos: um de M bytes e outro de N-M bytes;
- ... *first-fit* embora seja o algoritmo mais simples é rápido, pois finaliza a busca por um segmento o mais cedo possível, ou seja, não busca pela solução ótima.
- **algoritmo next-fit** - trabalhando da mesma forma que o *first-fit*, com a ressalva de que armazena a posição da lista do último buraco alocado e, assim, na próxima chamada começa a busca a partir do ponto que parou.

... 4.2.3 - Gerenciamento de Memória com Listas Encadeadas

- **algoritmo best-fit** - procura pela lista inteira e toma o buraco de tamanho mais próximo do tamanho do segmento requisitado (M bytes);
- ... é um algoritmo lento que cria na memória buracos pequenos que dificilmente serão alocados, no entanto, quando o segmento procurado é grande, este algoritmo melhora as chances de encontrar um segmento de tamanho adequado;



... 4.2.3 - Gerenciamento de Memória com Listas Encadeadas

- **algoritmo quick-fit** - mantém listas separadas para tamanhos requeridos, ou seja, tem-se um tabela com n entradas onde cada entrada corresponde a um ponteiro para a lista de buracos de tamanho m bytes, sendo que os ponteiros seguintes apontam para listas de tamanho múltiplo de m bytes;

- ... é um algoritmo rápido, embora requeira mecanismos para classificar os buracos por tamanho; determinar seus vizinhos para uma possível fusão quando, por exemplo, um processo termina ou é movido para o disco;

- ... se fusões não forem feitas, rapidamente a memória se fragmenta em um grande número de pequenos buracos não utilizáveis;

✯ Todos os 04 algoritmos podem ter sua performance melhorada mantendo-se listas separadas para processos e buracos, no entanto este aumento de velocidade na alocação promove a diminuição de velocidade quando se trata de liberar memória.

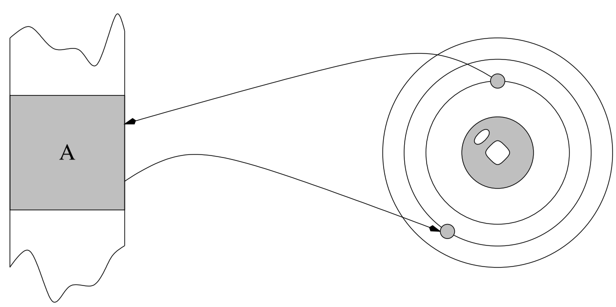
4.2.4 - Alocação de Espaço para Swapping

- Até agora, consideramos que a memória principal é rastreada quando processos retirados do disco, facilitando assim a alocação de espaço em memória;

- ... mas o que fazer em sistemas nos quais após um processo ter-se movido para a memória, nenhum espaço é mais reservado em disco para este processo ?

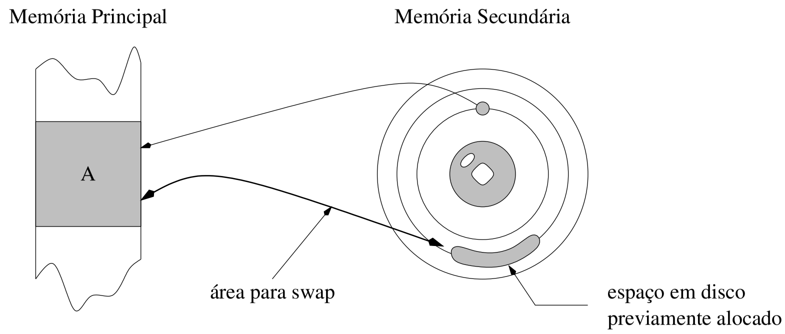
- ... quando do seu retorno ao disco, espaço deve ser alocado para abrigá-lo e, assim, em cada troca o processo poderá residir em diferentes regiões do disco;

Memória Principal Memória Secundária



... 4.2.4 - Alocação de Espaço para Swapping

- ... adicionalmente, os algoritmos para gerenciamento de espaço para permuta serão os mesmo usados para gerenciamento da memória principal;
- em outros sistemas, quando um processo é criado, um espaço para permuta é alocado em disco utilizando um dos algoritmos descritos acima;
- ... e, assim, sempre que um processo em memória dá lugar a outro processo, ele é colocado no espaço em disco alocado previamente a ele;



... 4.2.4 - Alocação de Espaço para Swapping

- ... quando um processo termina, o seu espaço para permuta é desalocado;
- Esta técnica é mais eficiente que a anterior pois uma única alocação de espaço em disco por processo é necessária permitindo que um possa processo possa ser permutado várias vezes durante sua execução;
- ... no entanto, uma área maior do disco deve ser reservada para *swapping*;