SO33B - Sistemas Operacionais

Parte 3 – Processos e Threads

Gerência de Memória

Matheus F. Mollon
04/11/2022

Sumário

- Introdução
- Funções básicas
- Alocação contígua simples
- Técnica de overlay
- Alocação Particionada
  - Alocação Particionada Estática
  - Alocação Particionada Dinâmica
  - Estratégias de Alocação de Partição
- Swapping

Contextualização

- Historicamente, a memória principal sempre foi vista como um recurso escasso e caro.

- Uma das maiores preocupações dos projetistas foi desenvolver sistemas operacionais que não ocupassem muito espaço de memória e, ao mesmo tempo, otimizassem a utilização dos recursos computacionais.

Contextualização

- Nos sistemas monoprogramáveis a gerência da memória não é muito complexa.

- Nos sistemas multiprogramáveis ela se torna crítica → necessidade de se maximizar o número de usuários e aplicações → utilização eficiente do espaço de memória principal.

Funções Básicas

- Os programas são armazenados em memórias secundárias, por serem um meio não volátil, de maior capacidade de armazenamento e de custo menor.

Armazenamento dos programas

SO transfere os programas

CPU busca as instruções para execução.

Funções Básicas

- O tempo de acesso à memória secundária é muito superior ao tempo de acesso à memória principal.
- O sistema operacional deve buscar reduzir o número de operações de E/S à memória secundária.
- caso contrário podem ser ocasionados sérios problemas no desempenho do sistema.

Funções Básicas

- Manter na memória principal o maior número de processos residentes.
- Mesmo na ausência de espaço livre, o sistema deve permitir que novos processos sejam aceitos e executados.
- Permitir a execução de programas que sejam maiores que a memória física disponível.

Funções Básicas

- O sistema operacional deve proteger as áreas de memória ocupadas por cada processo, além da área onde reside o próprio sistema.

- Mecanismos de compartilhamento devem ser oferecidos para que diferentes processos possam trocar dados de forma protegida.

Alocação Contígua Simples

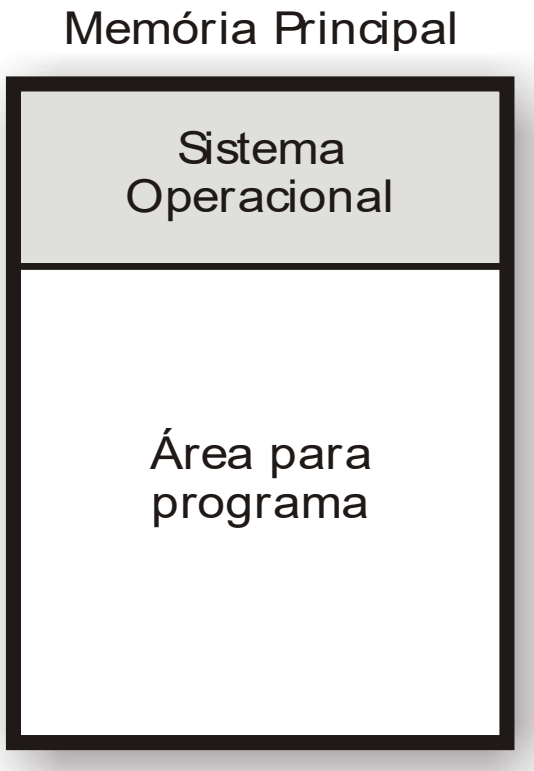
- Foi implementada nos primeiros sistemas operacionais, porém ainda está presente em alguns sistemas monoprogramáveis.

- Neste tipo de organização, a memória principal é subdividida em duas áreas: uma para o sistema operacional e outra para o programa do usuário.

- Dessa forma, o programador deve desenvolver suas aplicações preocupado, apenas, em não ultrapassar o espaço de memória disponível.

Alocação Contígua Simples

- Alocação Contígua Simples



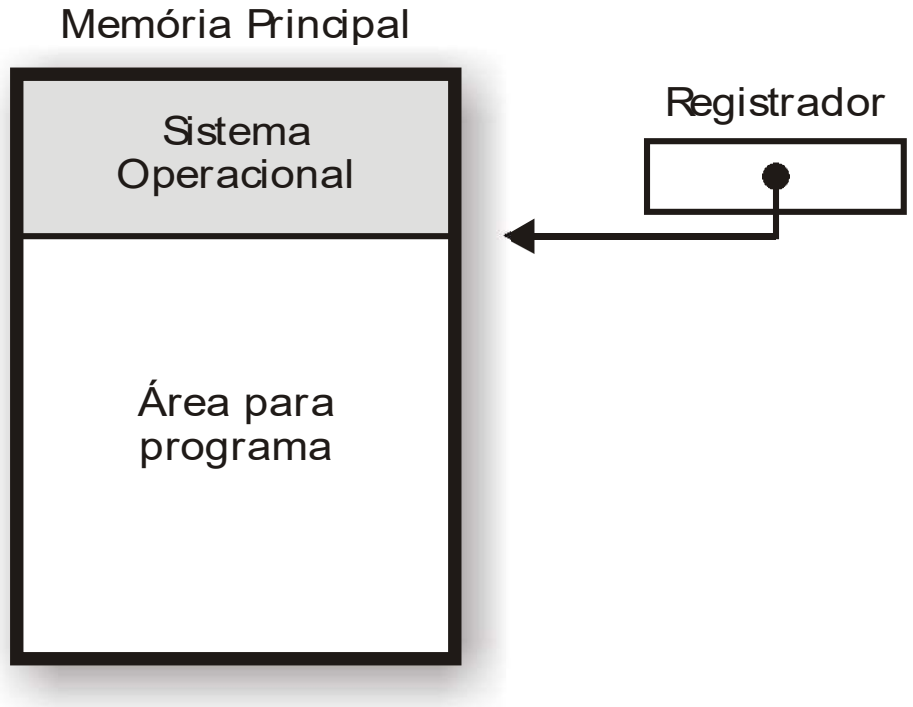
Alocação Contígua Simples

- O usuário tem controle sobre toda a memória principal, podendo ter acesso a qualquer posição de memória, inclusive a área do SO.

- Alguns sistemas implementam proteção através de um registrador que delimita as áreas do sistema operacional e do usuário.

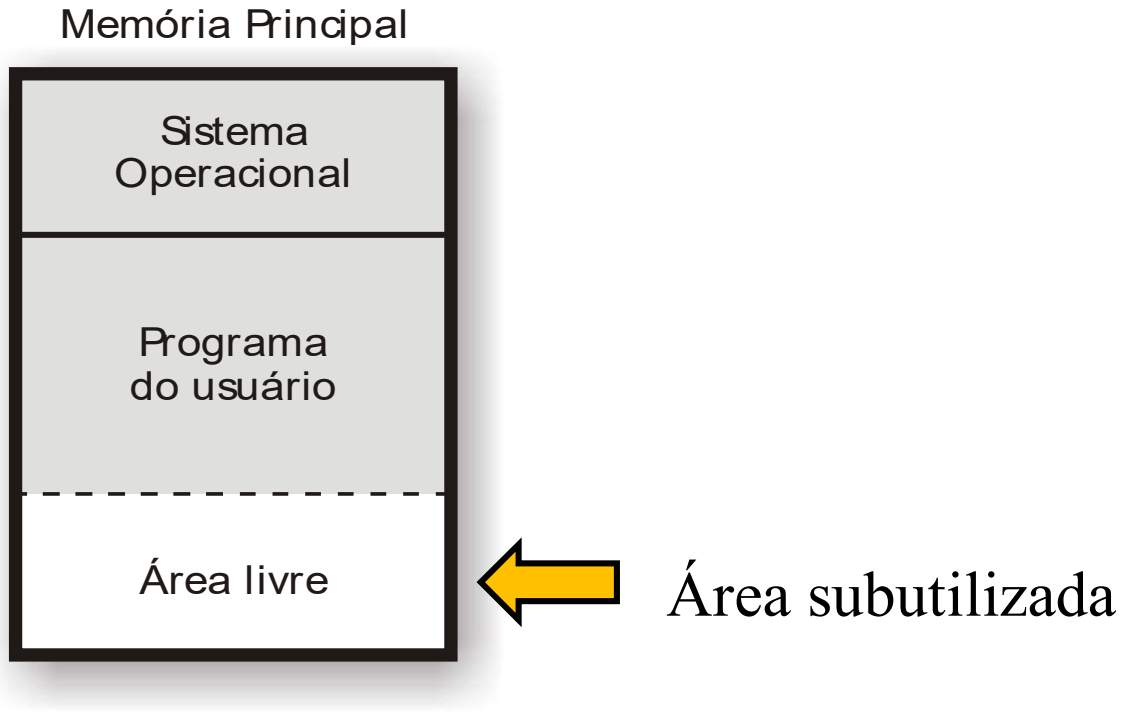
Alocação Contígua Simples

- Proteção



Alocação Contígua Simples

- Subutilização da memória



Técnica de Overlay

- Problema na alocação contígua simples: todos os programas estão limitados ao tamanho da área de memória principal disponível para o usuário.

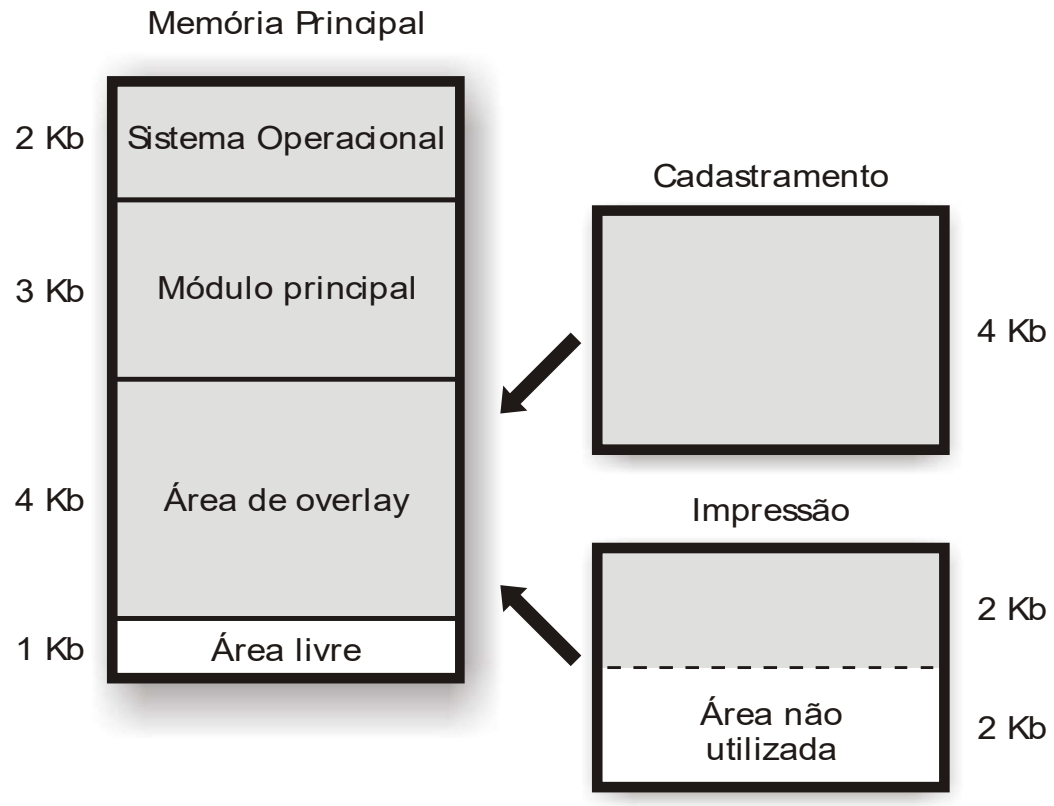
- Solução: dividir o programa em módulos, de forma que seja possível a execução independente de cada módulo, utilizando uma mesma área de memória.

Técnica de Overlay

- Essa técnica é chamada de overlay.
- A independência do código significa que quando um módulo estiver na memória para execução o outro não precisa necessariamente estar presente.
- A definição das áreas de overlay é função do programador.

Técnica de Overlay

- Técnica de Overlay



Técnica de Overlay

- O tamanho de uma área de overlay é estabelecido a partir do tamanho do maior módulo.

- A técnica de overlay tem a vantagem de permitir ao programador expandir os limites da memória principal.

- A utilização dessa técnica exige muito cuidado, pois pode trazer implicações tanto na sua manutenção quanto no desempenho das aplicações.

Alocação Particionada Estática

- Nos primeiros sistemas multiprogramáveis, a memória era dividida em pedaços de tamanho fixo, chamados partições.

- O tamanho das partições, estabelecido na fase de inicialização do sistema, era definido em função do tamanho dos programas que executariam no ambiente.

- Se fosse necessário alterar o tamanho de uma partição, o sistema deveria ser desativado e reinicializado.

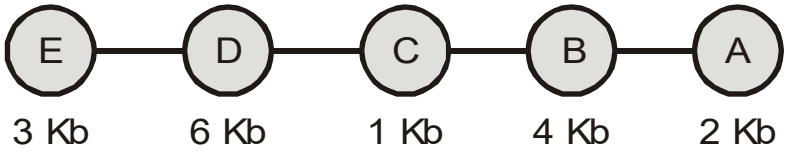
Alocação Particionada Estática

- Alocação Particionada Estática

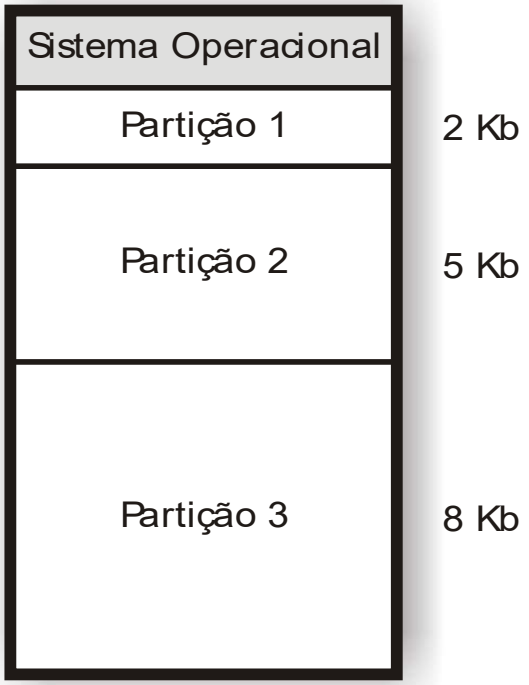
Tabela de partições

Partição	Tamanho
1	2 Kb
2	5 Kb
3	8 Kb

Programas a serem executados:



Memória Principal

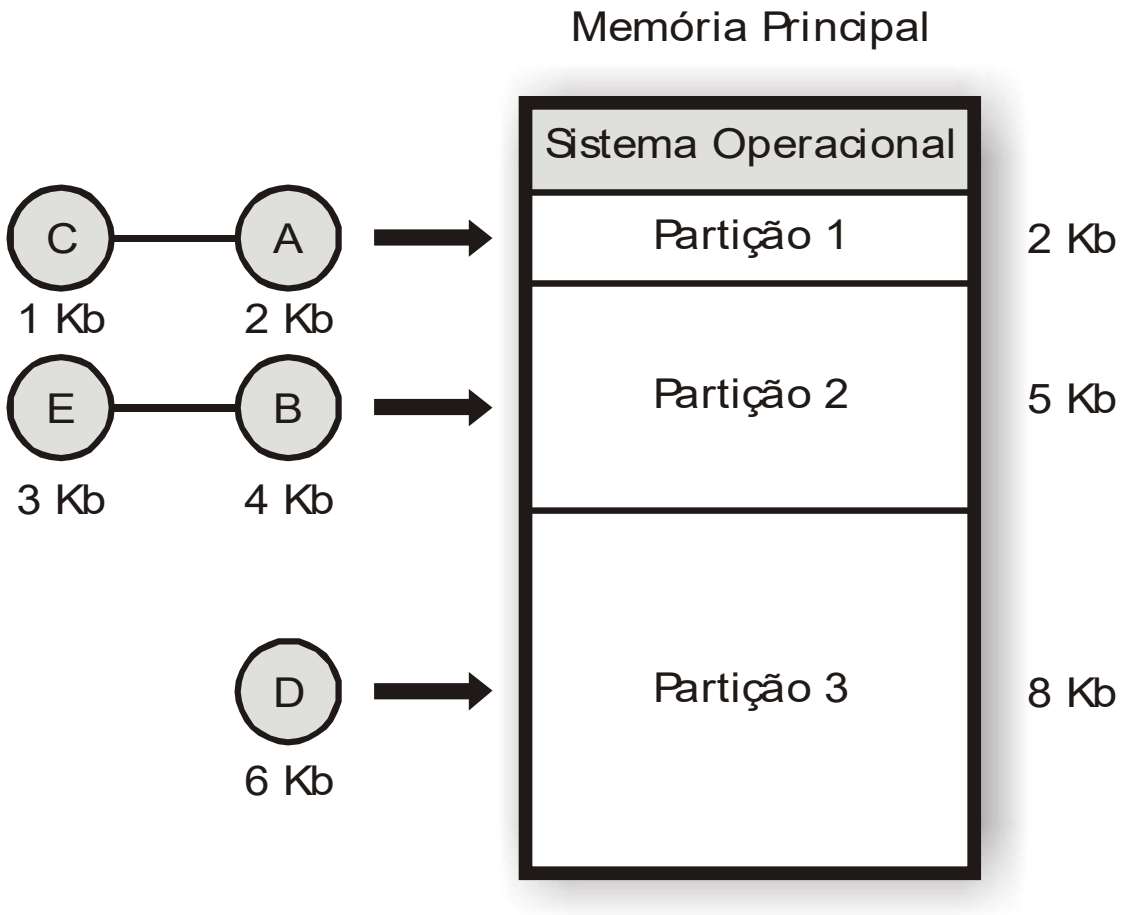


Alocação Particionada Estática

- A priori, os programas só poderiam ser carregados e executados em apenas uma partição específica.
- Limitação: compiladores e montadores geravam apenas o código absoluto.
- O programa só poderia ser carregado a partir do endereço de memória especificado no seu código.
- Este tipo de gerência chama-se alocação particionada estática absoluta.

Alocação Particionada Estática

- Alocação Particionada Estática Absoluta



Alocação Particionada Estática

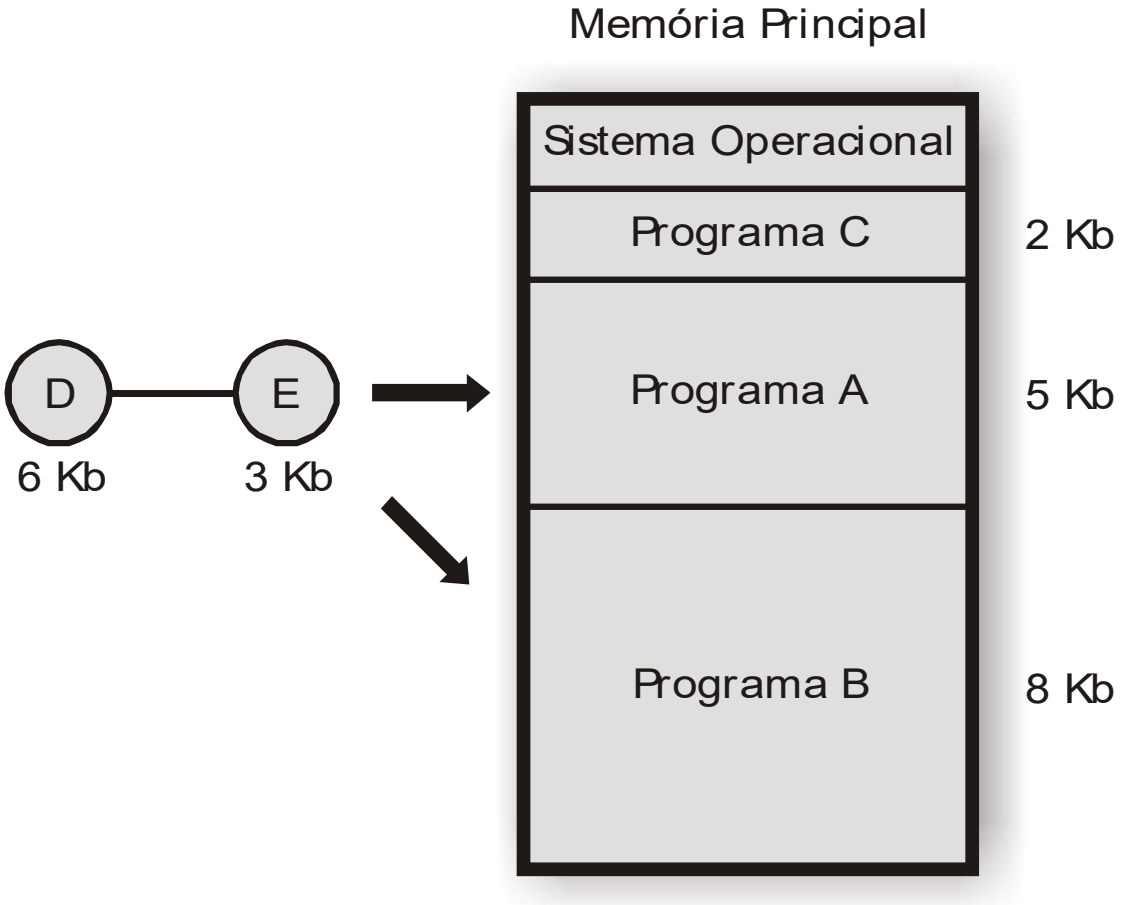
- Com a evolução dos compiladores, montadores, likers e loaders, o código gerado deixou de ser absoluto e passou a ser relocável.

- Código relocável: todas as referências a endereços no programa são relativas ao início do código, e não a endereços físicos de memória.

- Desta forma, os programas puderam ser executados de qualquer partição.

Alocação Particionada Estática

- Alocação Particionada Estática Relocável



Alocação Particionada Estática

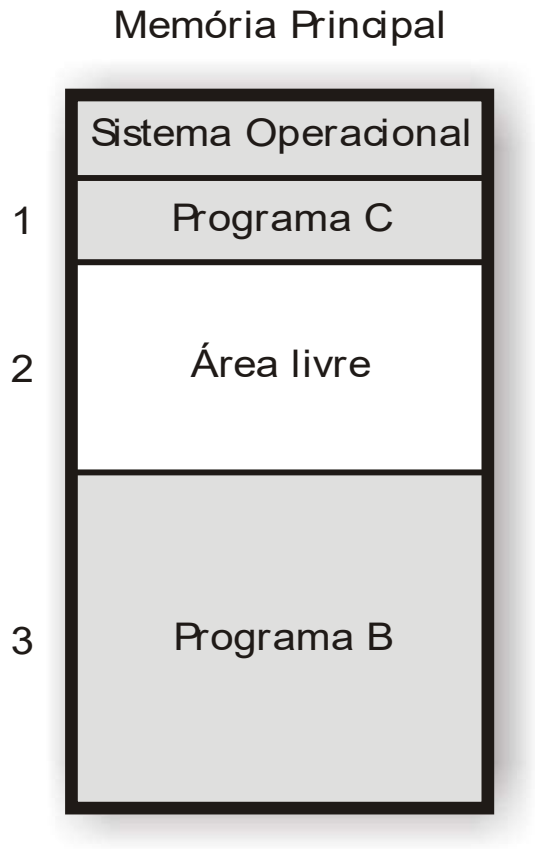
- Para manter o controle sobre quais partições estão alocadas, a gerência de memória mantém uma tabela de alocação de partições.

- Sempre que o programa é carregado para a memória, o sistema percorre a tabela visando encontrar uma posição livre para o programa ser carregado.

Alocação Particionada Estática

- Tabela de Alocação de Partições

Partição	Tamanho	Livre
1	2 Kb	Não
2	5 Kb	Sim
3	8 Kb	Não



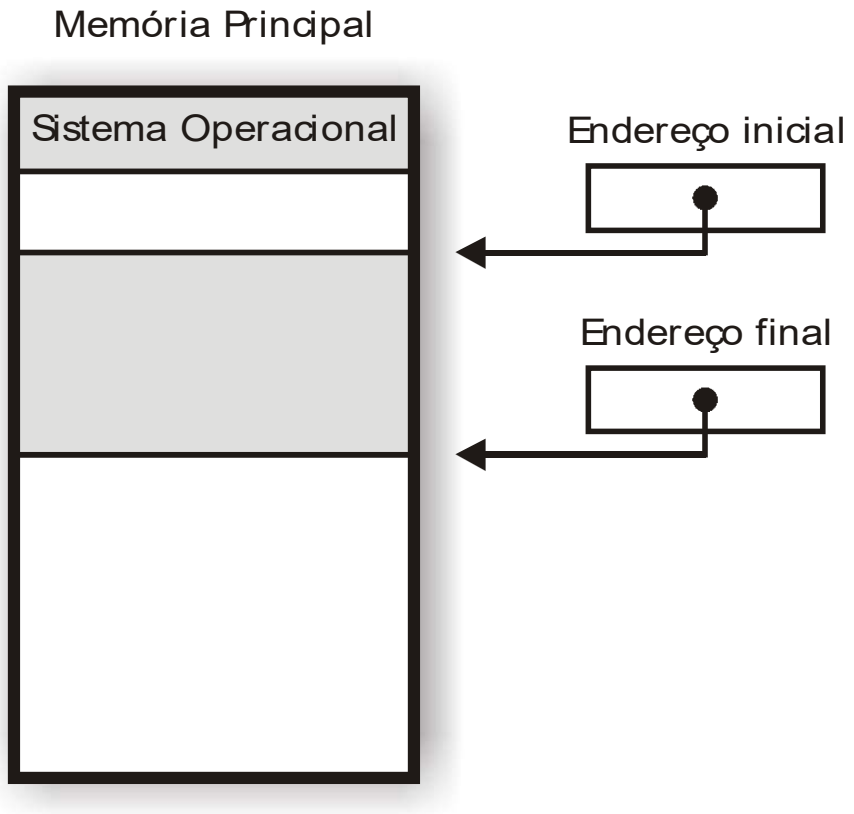
Alocação Particionada Estática

- Nesse esquema de alocação de memória a proteção baseia-se em dois registradores, que indicam os limites inferior e superior da partição onde o programa está sendo executado.

- Caso o programa tente acessar uma posição de memória fora dos limites definidos pelos registradores, ele é interrompido e uma mensagem de violação de acesso é gerada pelo sistema operacional.

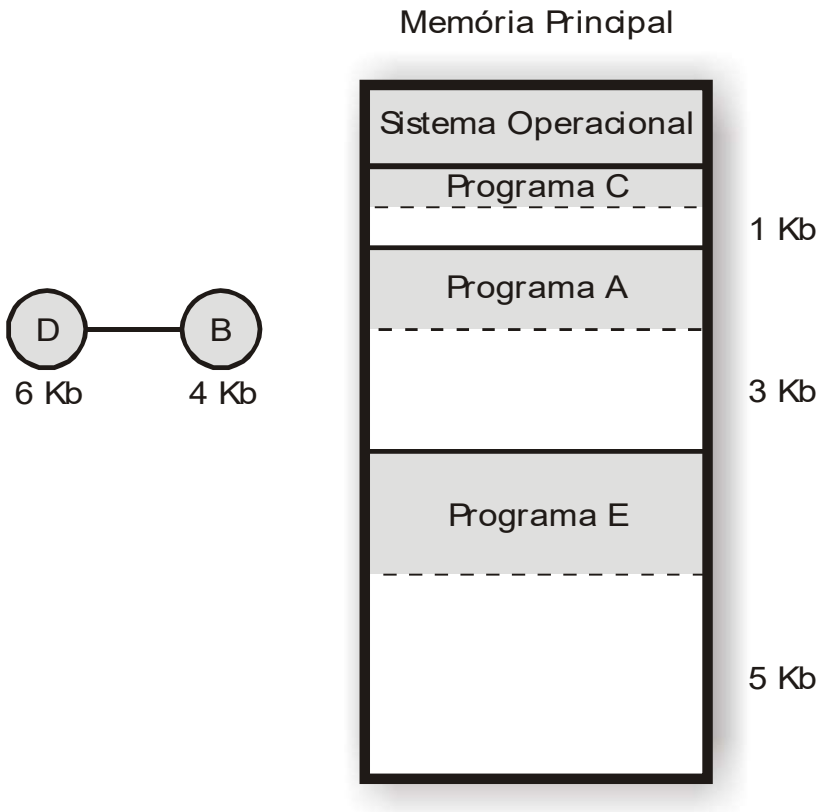
Alocação Particionada Estática

- Proteção



Alocação Particionada Estática

- Fragmentação Interna

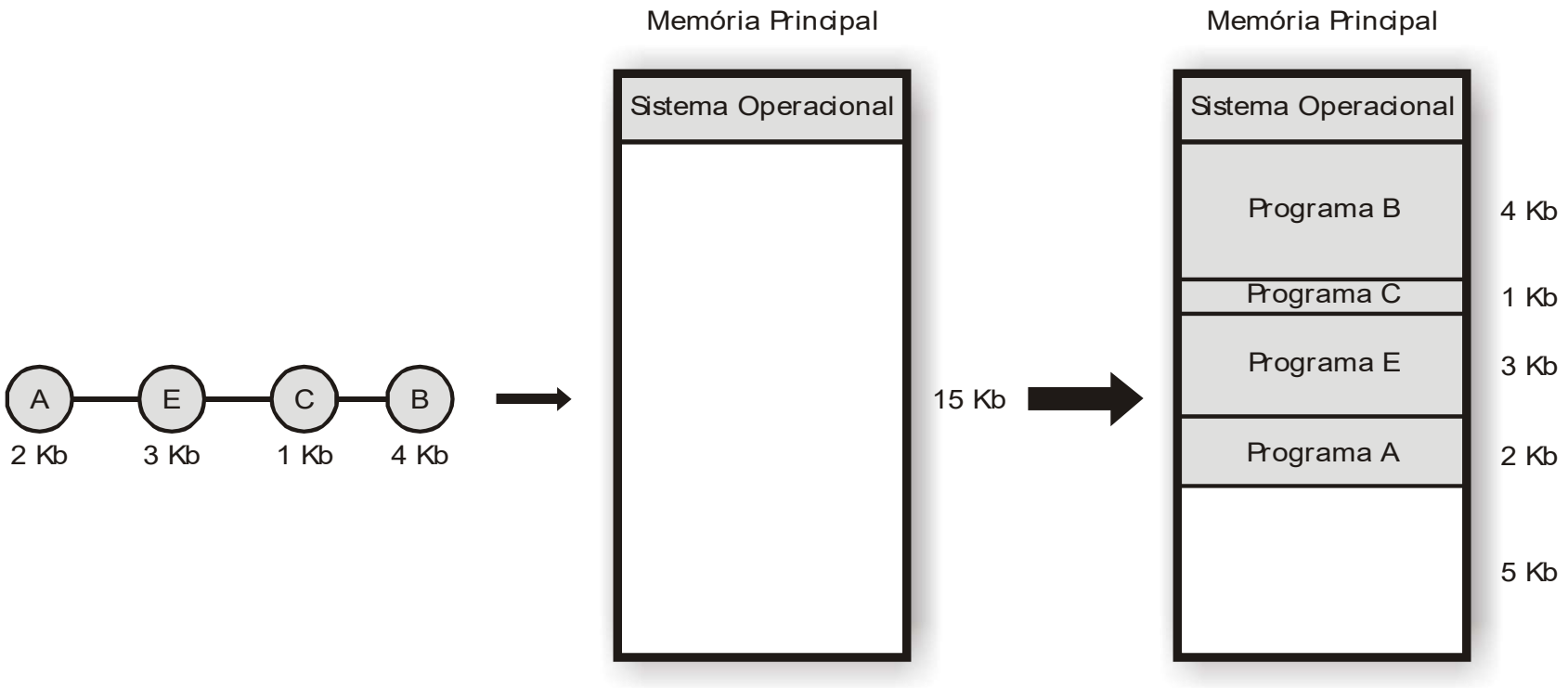


Alocação Particionada Dinâmica

- Neste tipo de gerenciamento foi eliminado o conceito de partições de tamanho fixo.
- Cada programa utilizaria o espaço necessário, tornando essa área sua partição.
- O problema da fragmentação interna não ocorre.
- Porém, há a ocorrência de fragmentação externa.

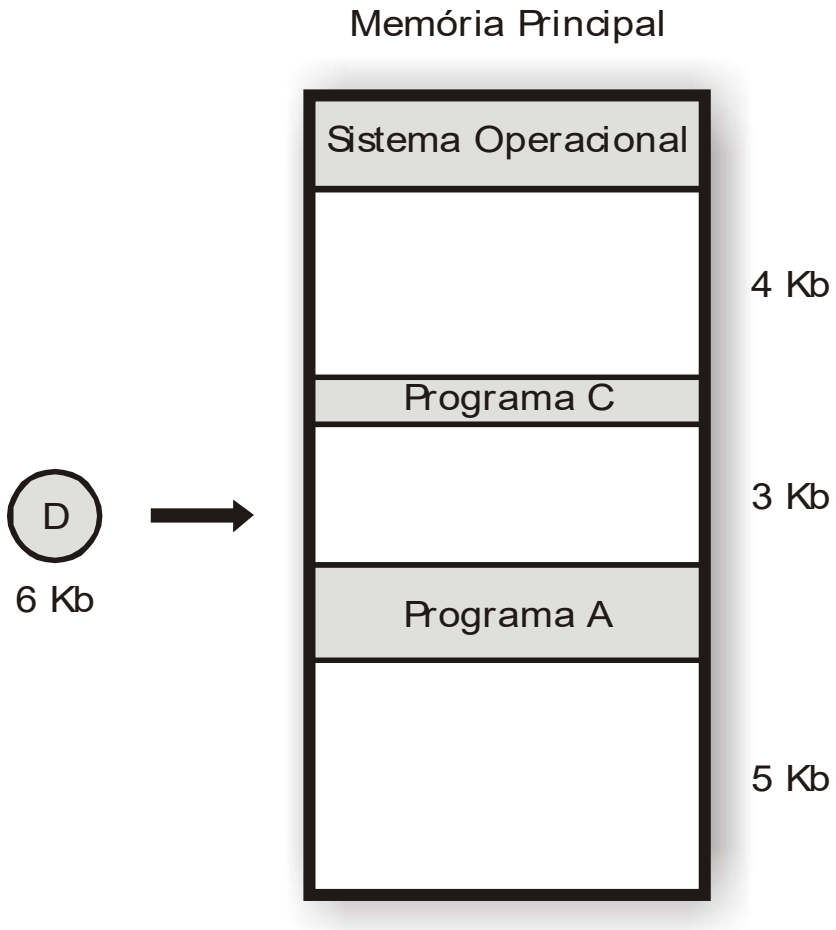
Alocação Particionada Dinâmica

- Alocação Particionada Dinâmica



Alocação Particionada Dinâmica

- Fragmentação Externa



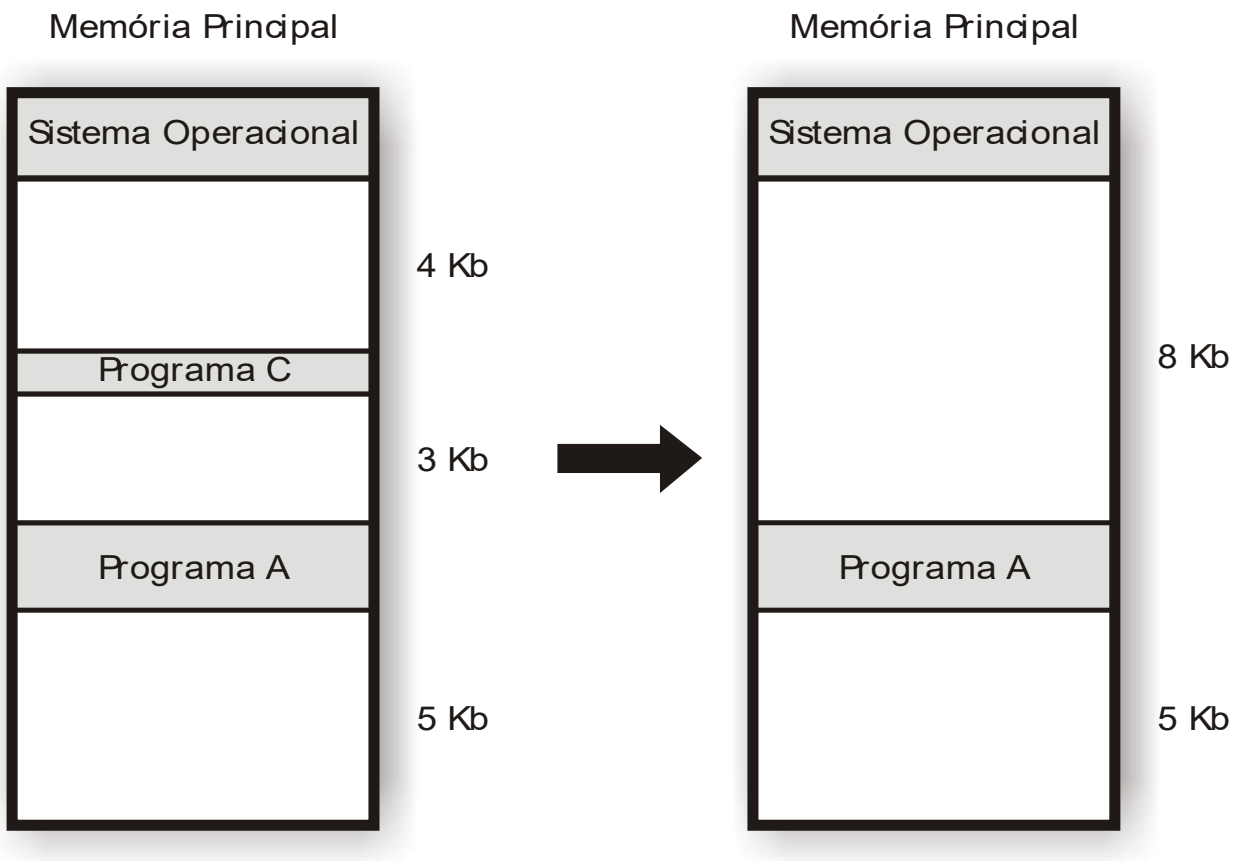
Alocação Particionada Dinâmica

- Existem duas soluções para o problema da fragmentação externa da memória principal.

- Na primeira solução, conforme os programas terminam apenas os espaços livres adjacentes são reunidos, produzindo áreas livres de tamanho maior.

Alocação Particionada Dinâmica

- Solução para a Fragmentação Externa



Alocação Particionada Dinâmica

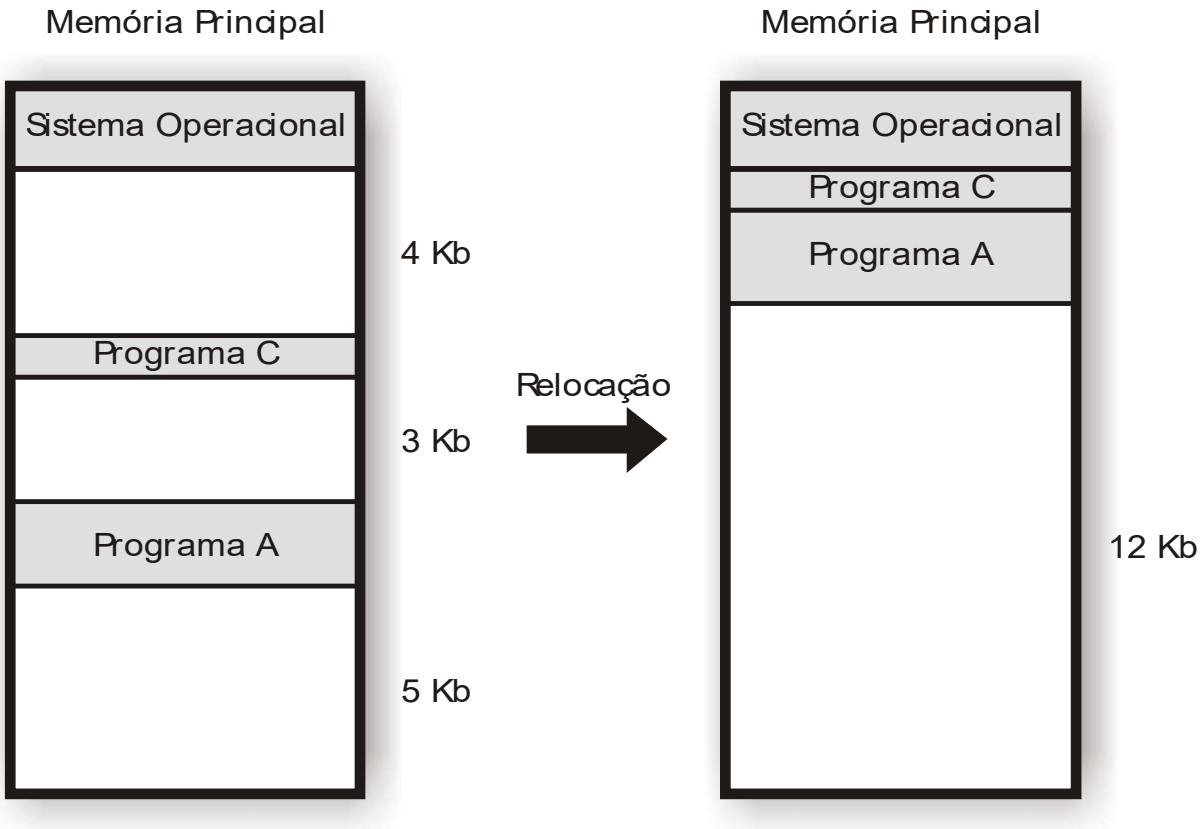
- Existem duas soluções para o problema da fragmentação externa da memória principal.

- A segunda solução envolve a relocação de todas as partições ocupadas, eliminando todos os espaços entre elas e criando uma única área livre contígua.

- Para que esse processo seja possível é necessário que o sistema tenha a capacidade de mover os diversos programas na memória principal, ou seja, realizar relocação dinâmica.

Alocação Particionada Dinâmica

- Solução para a Fragmentação Externa



Alocação Particionada Dinâmica

- Esse mecanismo de compactação → alocação particionada dinâmica com relocação.

- Reduz o problema da fragmentação, porém a complexidade do seu algoritmo e o consumo de recursos do sistema, como processador e área em disco, podem torná-lo inviável.

Estratégias de Alocação

- Os sistemas operacionais implementam, basicamente, três estratégias para determinar em qual área livre um programa será carregado para execução.

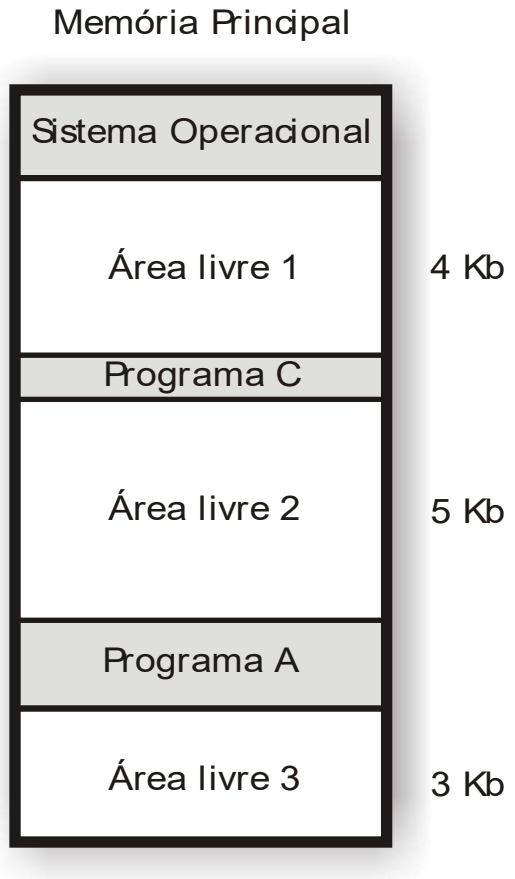
- Essas estratégias tentam evitar ou diminuir o problema da fragmentação externa.

- A melhor estratégia a ser adotada por um sistema depende de uma série de fatores, sendo o mais importante o tamanho dos programas processados no ambiente.

Estratégias de Alocação

- Lista de Áreas Livres

Áreas livres	Tamanho
1	4 Kb
2	5 Kb
3	3 Kb

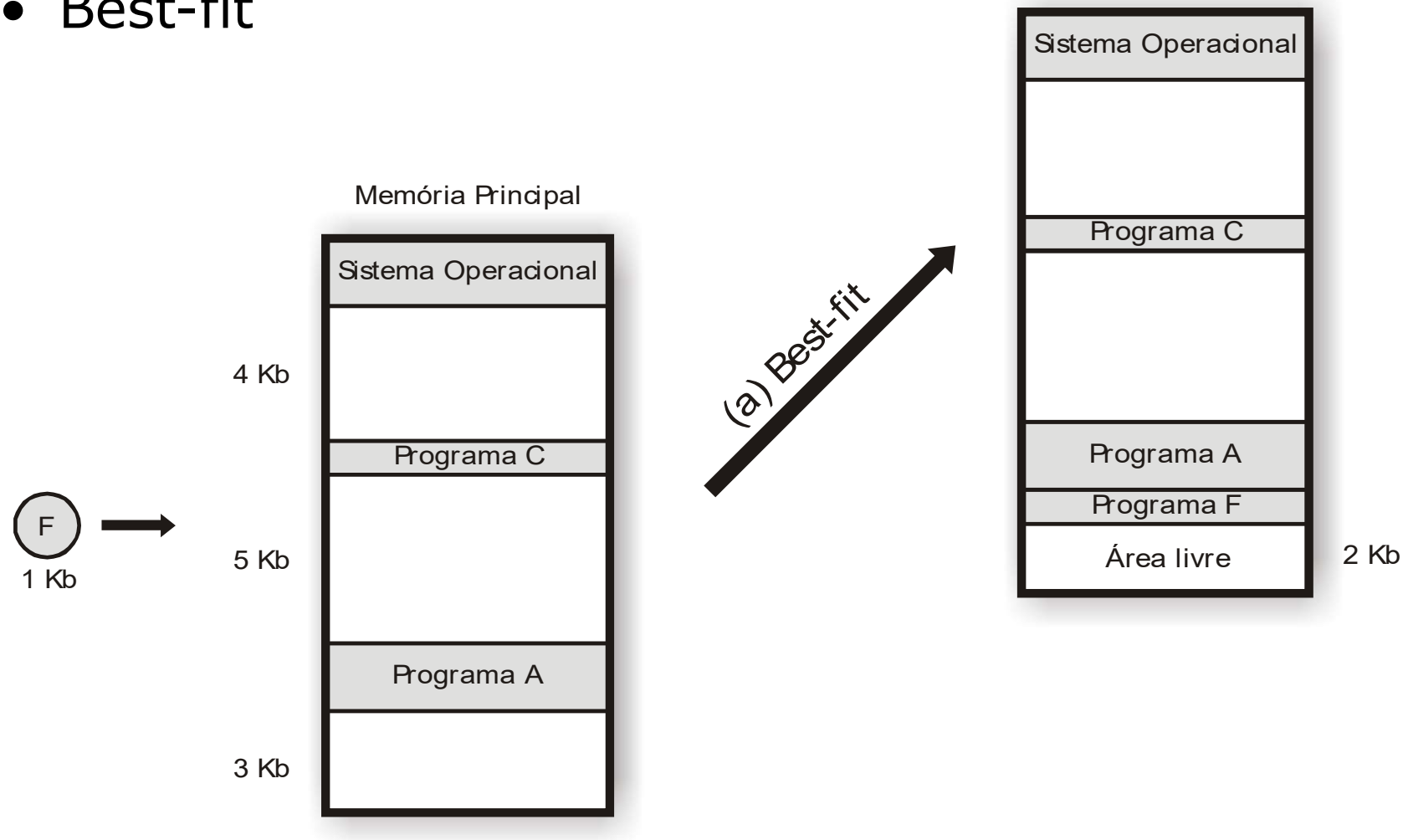


Estratégias de Alocação

- Best-fit:
  - A melhor partição é escolhida, ou seja, aquela em que o programa deixa o menor espaço sem utilização.
  - A lista de áreas livres está ordenada por tamanho, diminuindo o tempo de busca por uma área desocupada.

Estratégias de Alocação

- Best-fit



Estratégias de Alocação

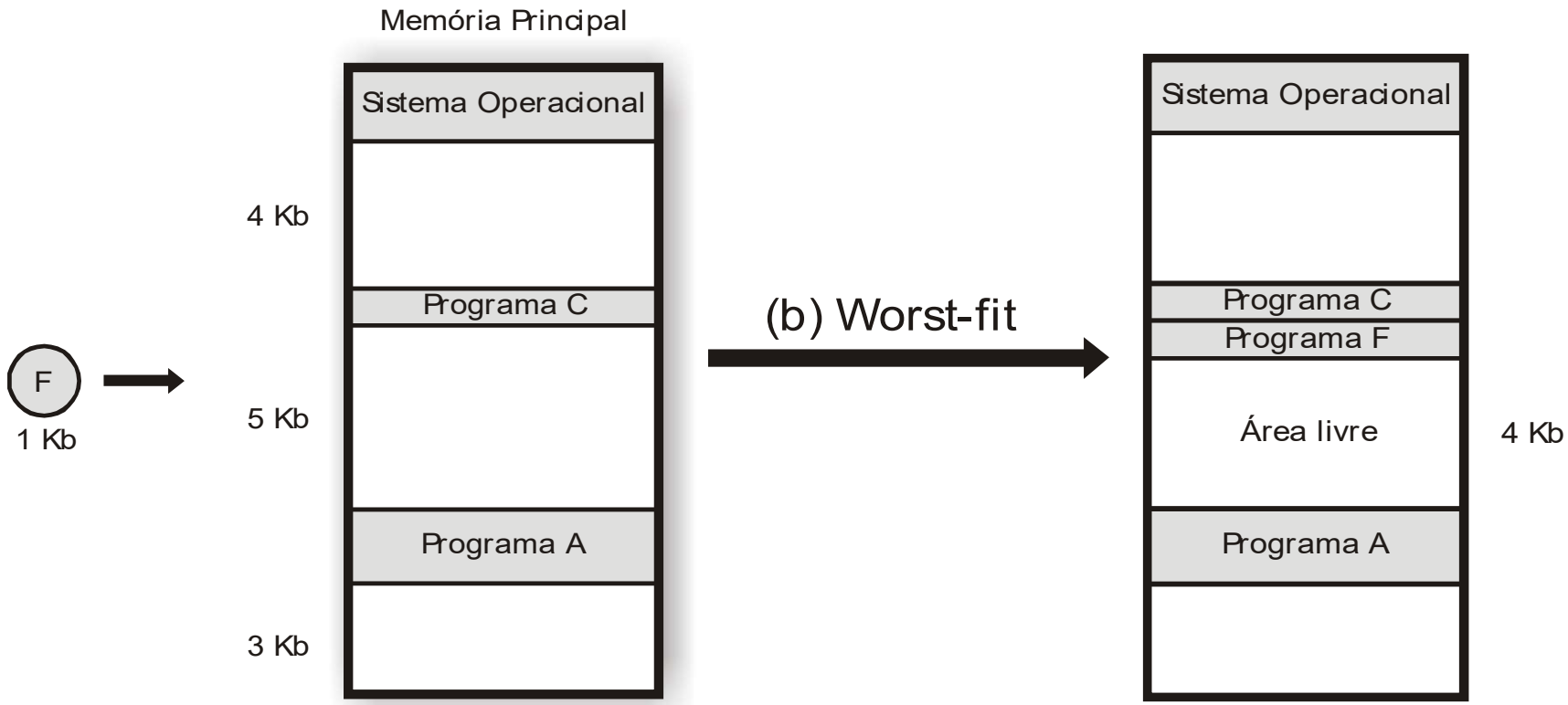
- Best-fit:
  - A melhor partição é escolhida, ou seja, aquela em que o programa deixa o menor espaço sem utilização.
  - A lista de áreas livres está ordenada por tamanho, diminuindo o tempo de busca por uma área desocupada.
  - Desvantagem: Como é alocada a partição que deixa a menor área livre, a tendência é que cada vez mais a memória fique com pequenas áreas não contíguas → problema da fragmentação.

Estratégias de Alocação

- Worst-fit:
  - A pior partição é escolhida, ou seja, aquela em que o programa deixa o maior espaço sem utilização.
  - Apesar de utilizar as maiores partições, a técnica de worst-fit deixa espaços livres maiores que permitem a um maior número de programas utilizar a memória, diminuindo o problema da fragmentação.

Estratégias de Alocação

- Worst-fit

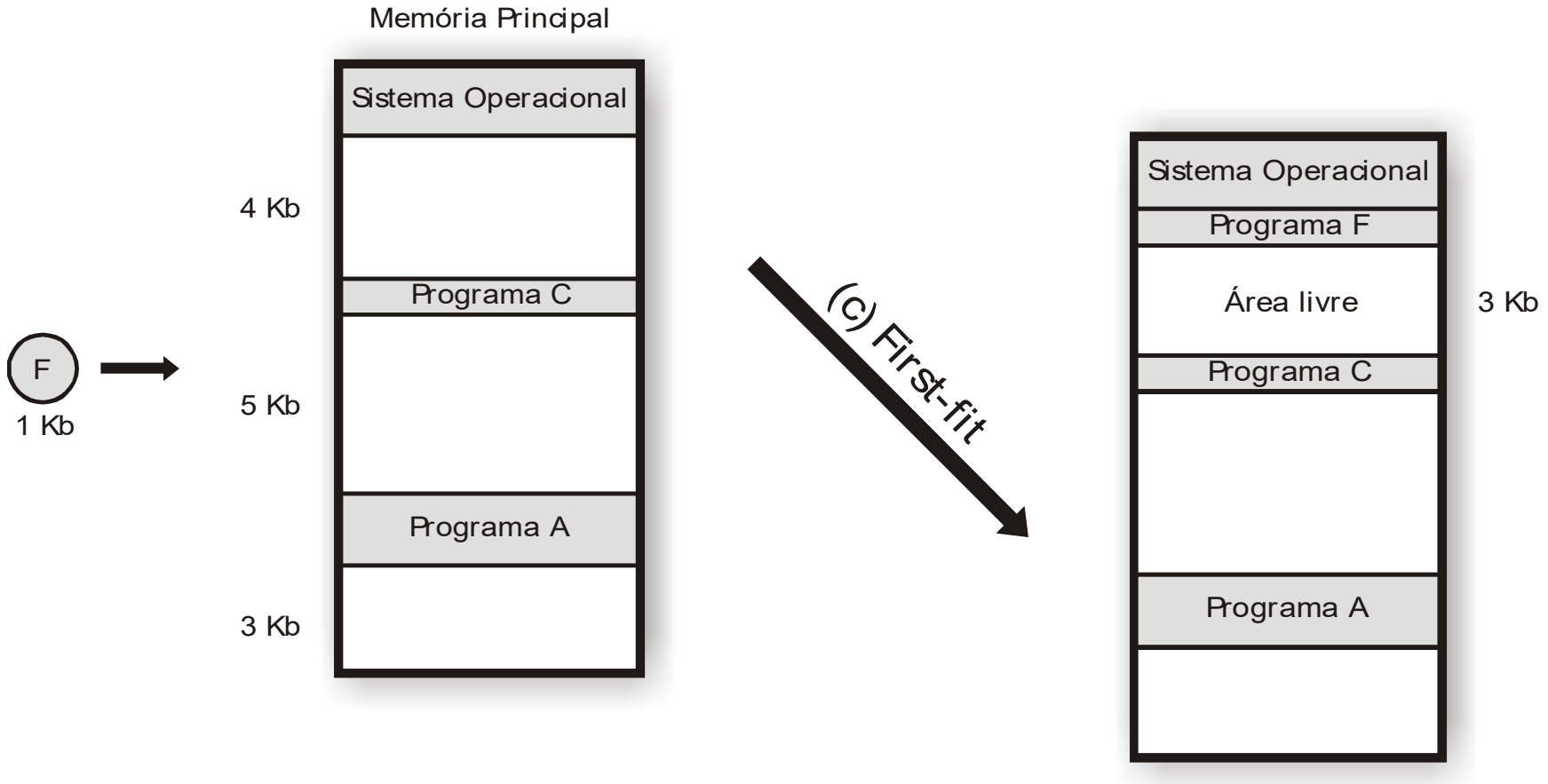


Estratégias de Alocação

- First-fit:
  - A primeira partição livre de tamanho suficiente para carregar o programa é escolhida.
  - A lista de áreas livres está ordenada por endereços crescentemente.
  - Há uma grande chance de se obter uma grande partição livre nos endereços de memória mais altos.
  - Esta é a estratégia mais rápida, consumindo menos recursos do sistema.

Estratégias de Alocação

- First-fit



Swapping

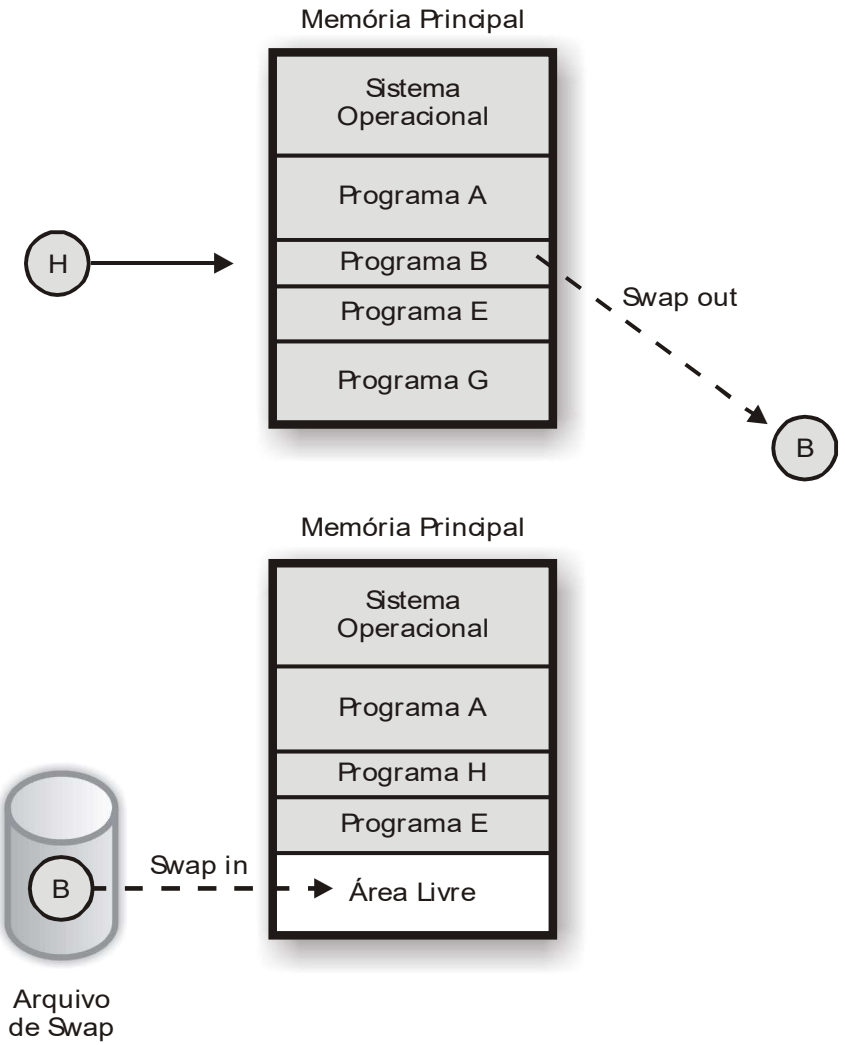
- É uma técnica aplicada à gerência de memória para programas que esperam por memória livre para serem executados.

- Swap out: o sistema escolhe um processo residente, que é transferido da memória principal para a memória secundária.

- Swap in: o processo é carregado de volta da memória secundária para a memória principal e pode continuar sua execução como se nada tivesse ocorrido.

Swapping

- Swapping



Swapping

- O conceito de swapping permite um maior compartilhamento da memória principal.
- Seu maior problema é o elevado custo das operações de entrada/saída (swap in/out).
- Thrashing: em momentos em que há pouca memória disponível, o sistema pode ficar quase que dedicado à realização de swapping, deixando de executar outras tarefas.
- A técnica de swapping está presente na gerência de memória virtual.