# Gerenciamento de Memória

**Prof. Dr. José Luís Zem**
**Prof. Dr. Renato Kraide Soffner**
**Prof. Ms. Rossano Pablo Pinto**


**Faculdade de Tecnologia de Americana**
**Centro Paula Souza**

# Tópicos

- Introdução
- Alocação Contígua Simples
- Alocação Particionada
  - Alocação Particionada Estática
  - Alocação Particionada Dinâmica
  - Estratégias para a Escolha da Partição
- Swapping

# Introdução

- Na memória principal residem todos os programas e dados que serão executados ou referenciados pelo processador.

- Toda vez que deseja-se executar um programa residente na memória secundária, deve-se, de alguma forma, carregá-lo para a memória principal.

- Organização e gerência de memória principal têm sido fatores importantes no projeto de sistemas operacionais.

# Introdução

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

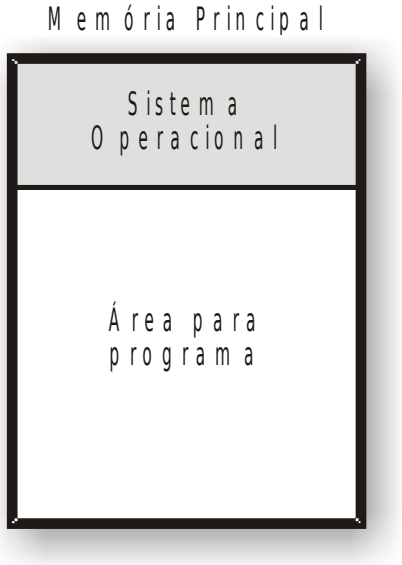
- Uma das maiores preocupações dos projetistas era desenvolver sistemas operacionais que não ocupassem muita memória e, ao mesmo tempo, otimizassem a sua utilização.

- Enquanto nos sistemas monoprogramáveis a gerência de memória não é muito complexa, nos sistemas multiprogramáveis ela se torna crítica.

# Alocação Contígua Simples

- A alocação contígua simples foi implementada nos primeiros sistemas operacionais desenvolvidos, porém, ainda está presente em alguns sistemas monoprogramáveis.

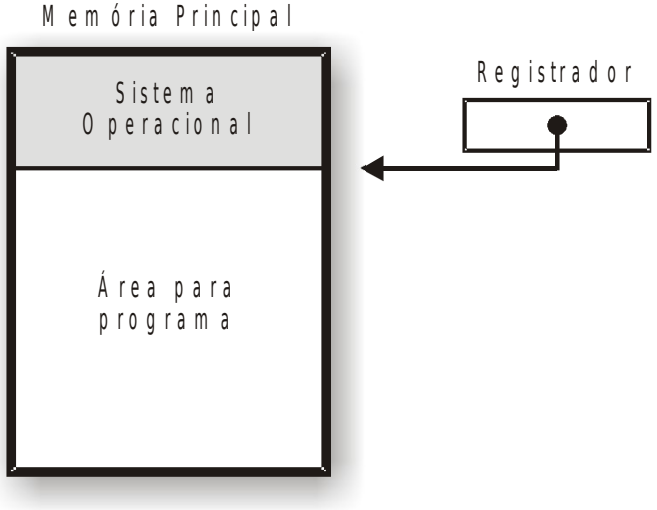
- Nesse tipo de organização, a memória principal é dividida em duas partes:
  - uma para o sistema operacional;
  - outra para o programa do usuário.



# Alocação Contígua Simples

- O usuário tem controle sobre toda a memória principal, podendo ter acesso a qualquer posição de memória, inclusive alterar e destruir o sistema operacional.

- Para protegê-lo desses ataques, conscientes ou não, alguns sistemas operacionais implementam proteção através de um registrador, que delimita as áreas do sistema operacional e do usuário.

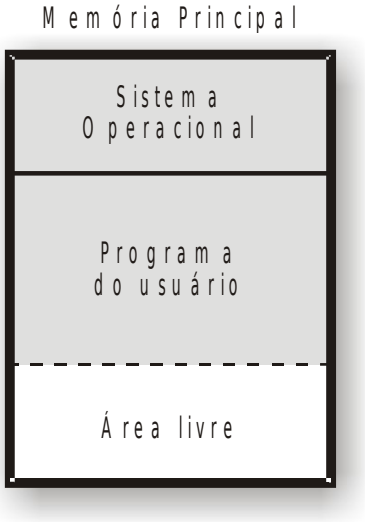


# Alocação Contígua Simples

- Apesar de sua simplicidade de implementação e código reduzido, a alocação contígua simples não permite a utilização eficiente do processador e da memória pois apenas um usuário pode utilizar este recurso.

- Em relação à memória, caso o programa não a preencha totalmente, existirá um espaço de memória sem utilização.

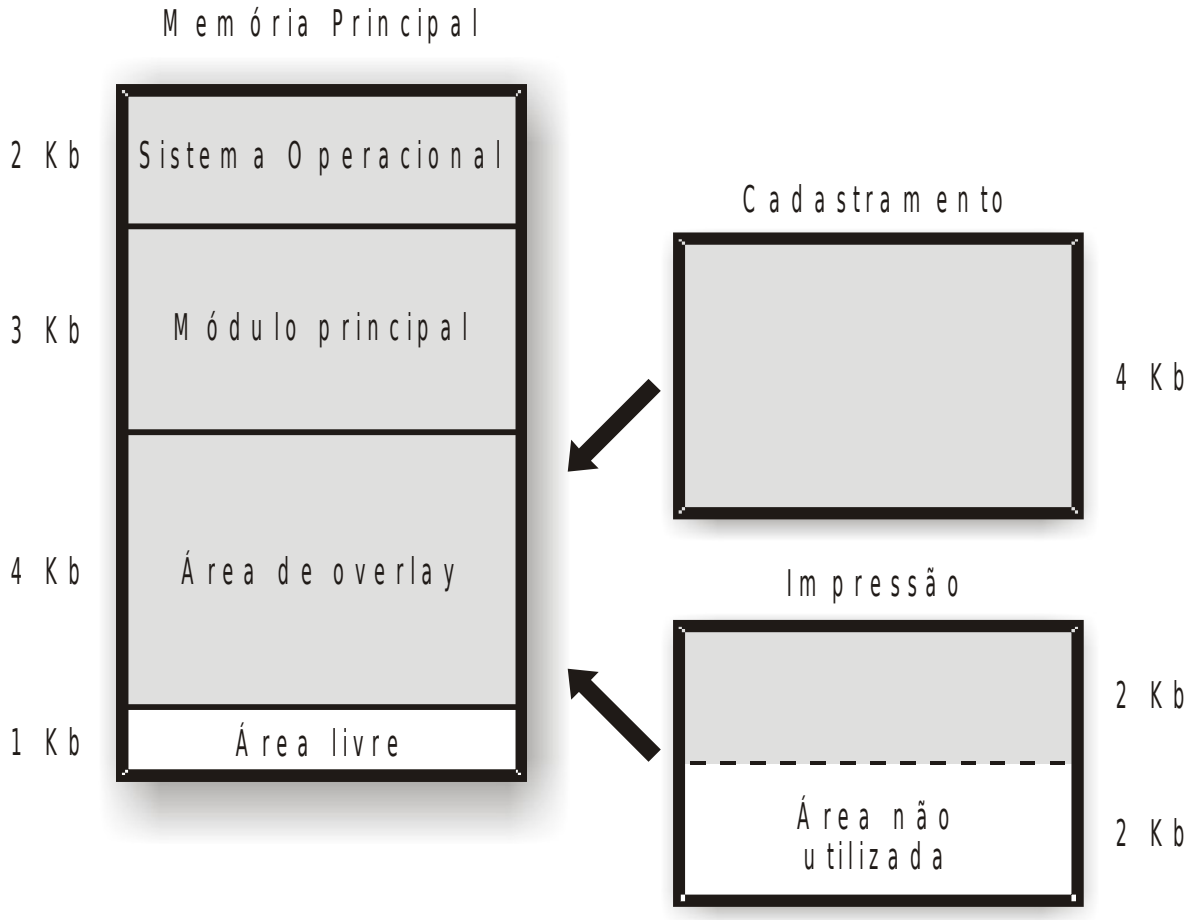
- Fragmentação.



# Alocação Contígua Simples

- No princípio, os programas dos usuários estavam limitados ao tamanho da memória principal disponível.
- A solução encontrada para o problema foi dividir o programa em partes (módulos), de forma que pudessem executar independentemente uma da outra, utilizando a mesma área da memória.
- Esta técnica recebeu o nome de overlay.

# Alocação Contígua Simples



# Alocação Particionada

- Os sistemas monoprogramáveis permitem que o processador permaneça ocioso e que a memória seja subutilizada, enquanto um programa aguarda o término de uma operação de I/O, por exemplo.

- A multiprogramação vem resolver este problema, pois enquanto aguarda algum evento, outros processos podem ser executados pela CPU nesse intervalo de tempo.

- Para a multiprogramação ser eficiente, é necessário que vários programas estejam na memória ao mesmo tempo, daí a necessidade de uma nova forma de organização da memória principal.

# Alocação Particionada Estática

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

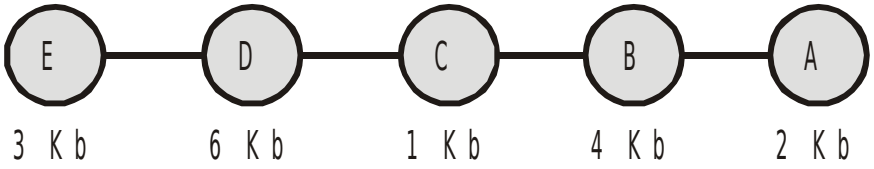
- O tamanho das partições eram estabelecidos na fase de inicialização do sistema (boot), em função do tamanho dos programas que seriam executados no ambiente.

# Alocação Particionada Estática

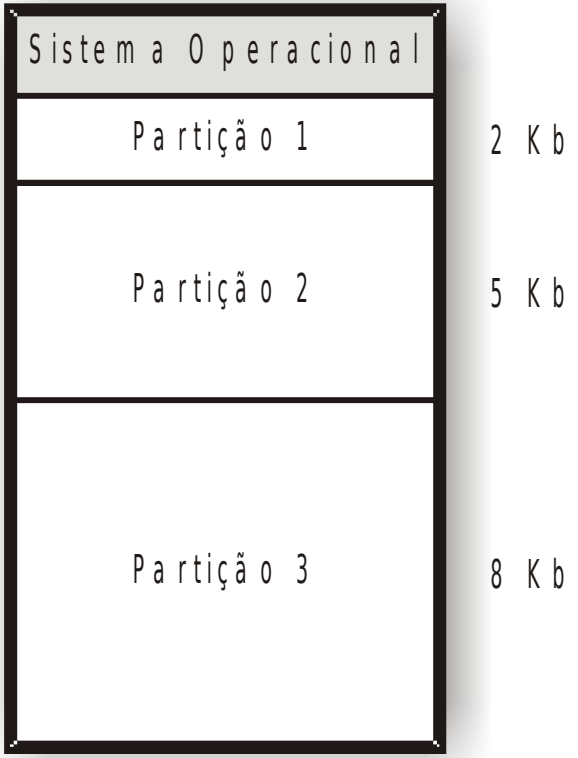
Tabela de partições

| Partição | Tamanho |
|---|---|
| 1 | 2 Kb |
| 2 | 5 Kb |
| 3 | 8 Kb |

Programas a serem executados:



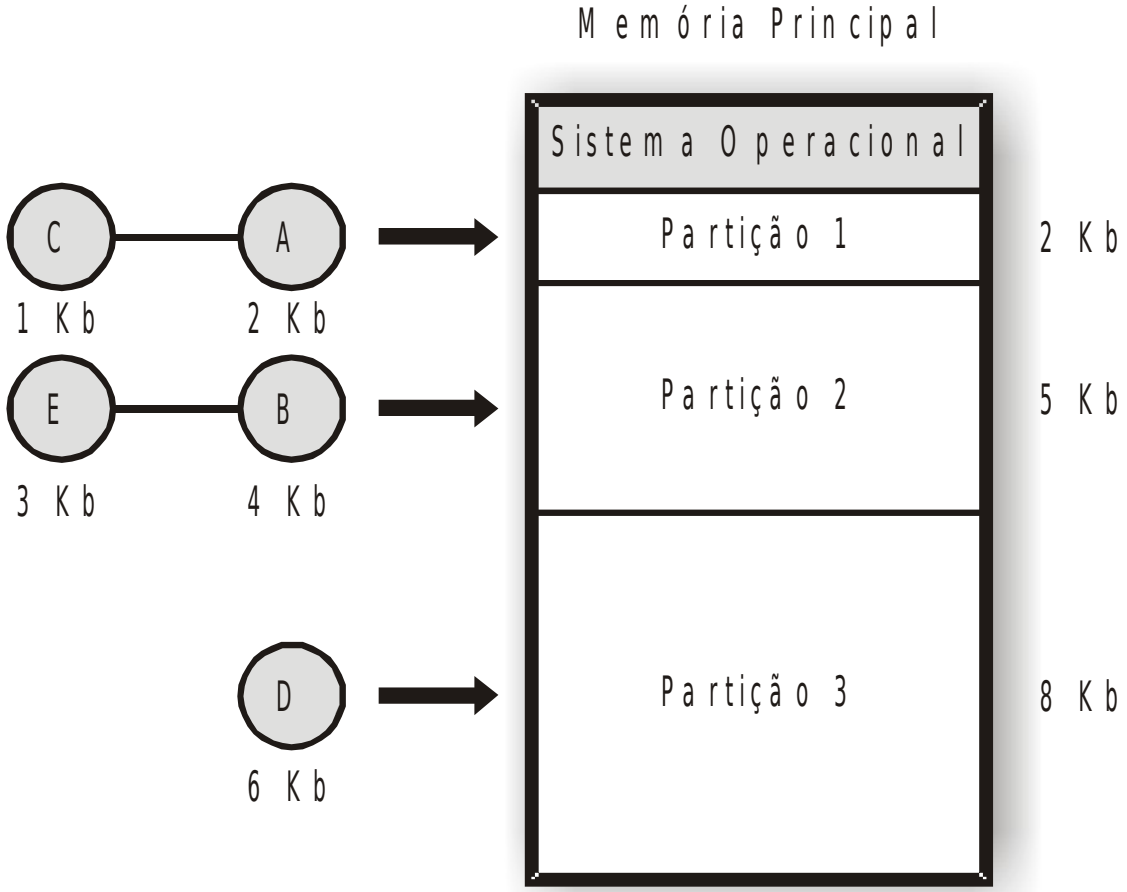

Memória Principal



# Alocação Particionada Estática

- No princípio, os programas só podiam ser executados em uma das partições, mesmo que outras estivessem disponíveis.
- Essa limitação era devido aos compiladores utilizados na época, que geravam códigos absolutos.
- Este tipo de alocação recebeu o nome de alocação particionada estática absoluta.

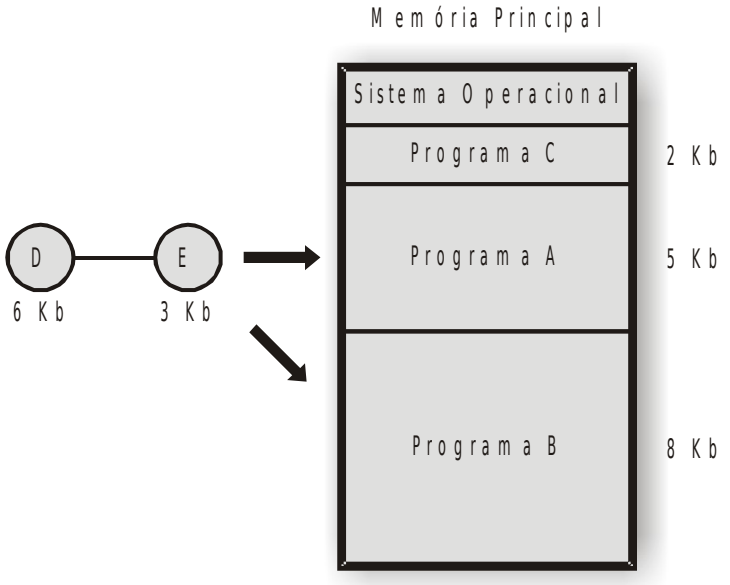
# Alocação Particionada Estática



# Alocação Particionada Estática

- Com a evolução dos compiladores, linkers e loadres, a geração de código relocável foi possível e os programas puderam ser carregados em qualquer partição.

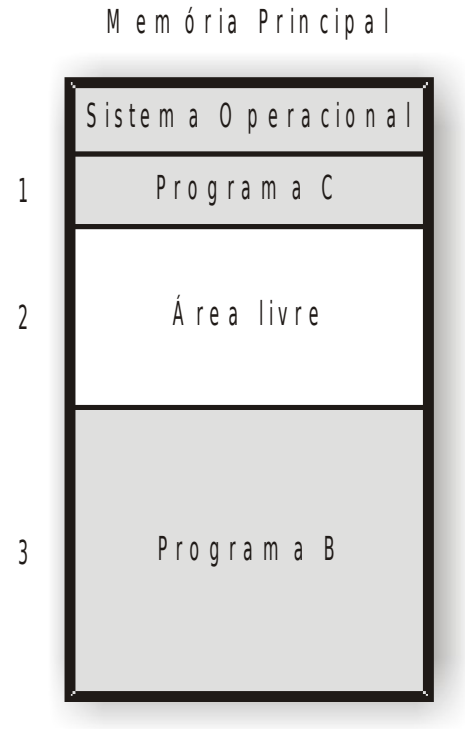
- A esse novo tipo de alocação deu-se o nome de alocação particionada estática relocável.



# Alocação Particionada Estática

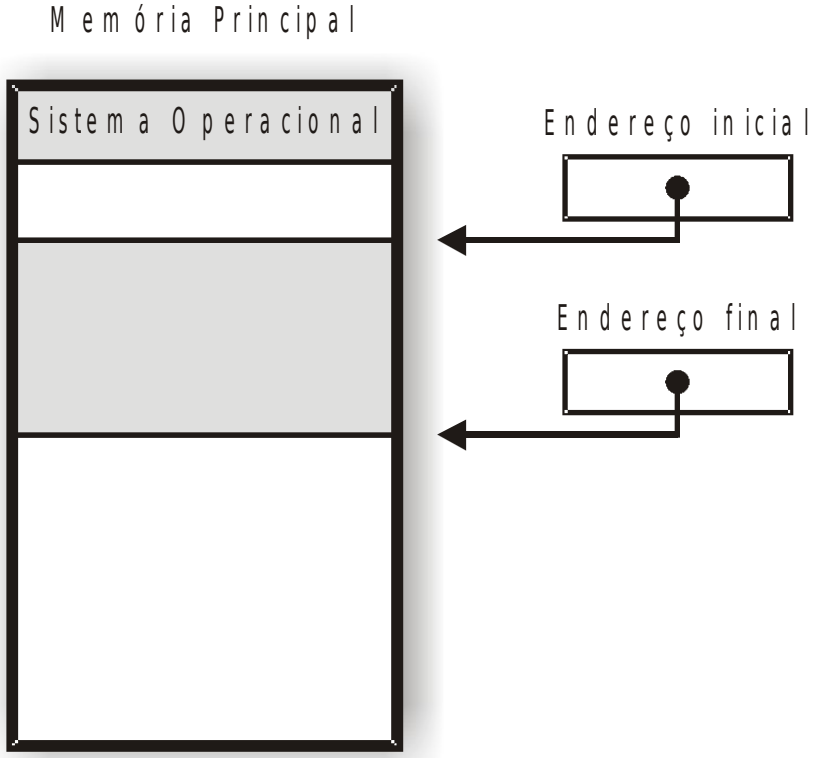
- Para manter o controle sobre quais as partições estavam alocadas ou não, os sistemas possuíam uma tabela delimitando cada partição, seu tamanho e se estava em uso ou não.

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



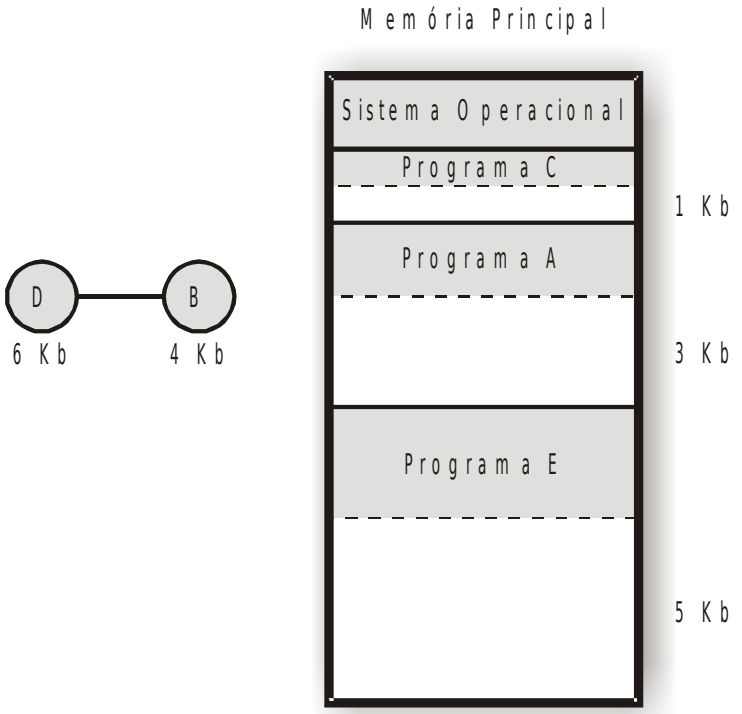
# Alocação Particionada Estática

- A proteção baseava-se em dois registradores, que indicavam os limites inferiores e superiores da partição onde o processo seria carregado.



# Alocação Particionada Estática

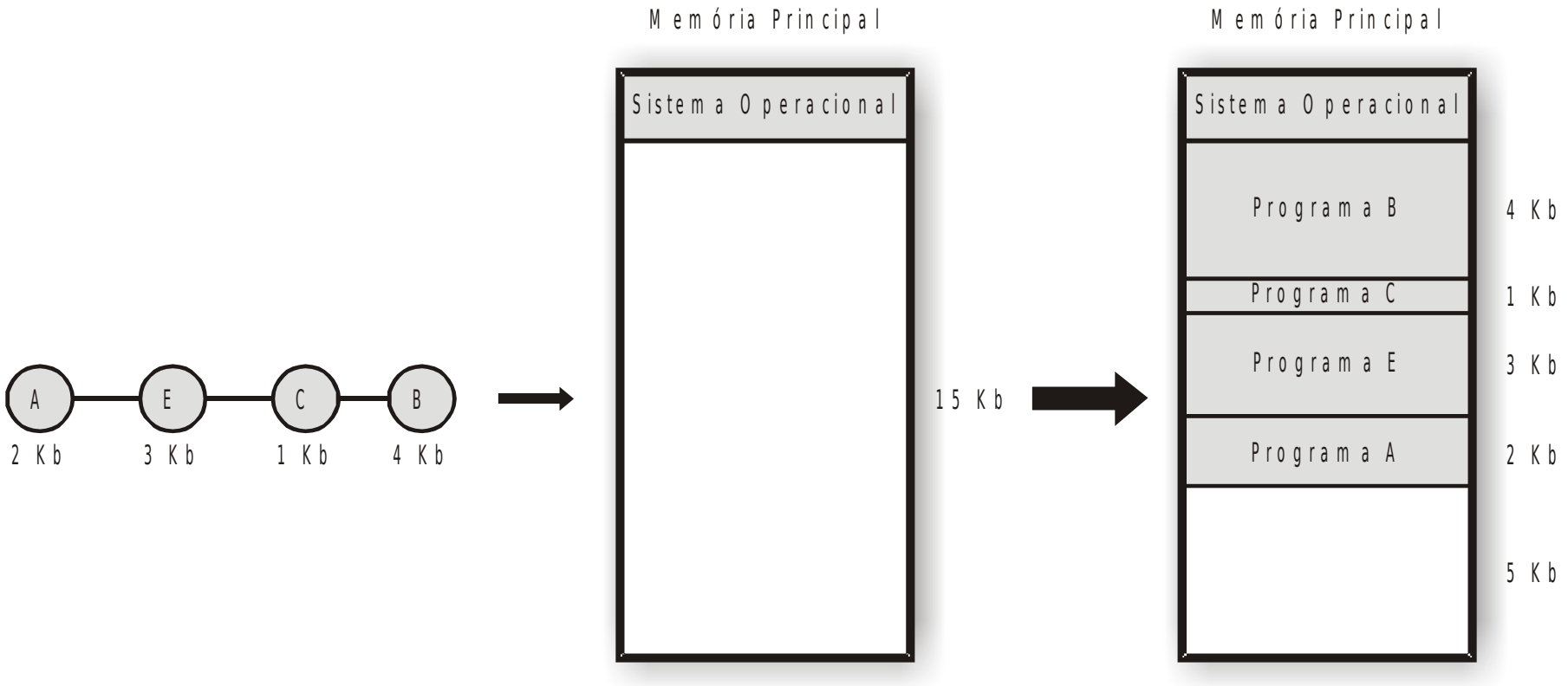
- Tanto nos sistemas de alocação absoluta como nos sistemas com alocação relocável, os processos não preenchiam totalmente as partições onde eram alocadas. Desta forma, a fragmentação também estava presente nestes esquemas.



# Alocação Particionada Dinâmica

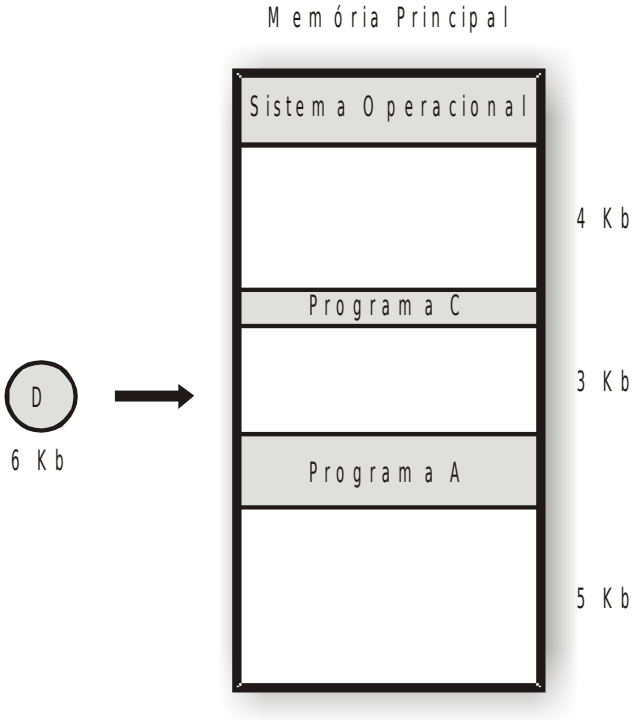
- Aumenta o grau de compartilhamento de memória.
- Diminui o grau de fragmentação.
- Neste esquema foi eliminado o conceito de partições com tamanho fixo.
- Cada programa utiliza o espaço que necessitasse, desde que existisse este espaço na memória,transformando-o em uma partição.

# Alocação Particionada Dinâmica



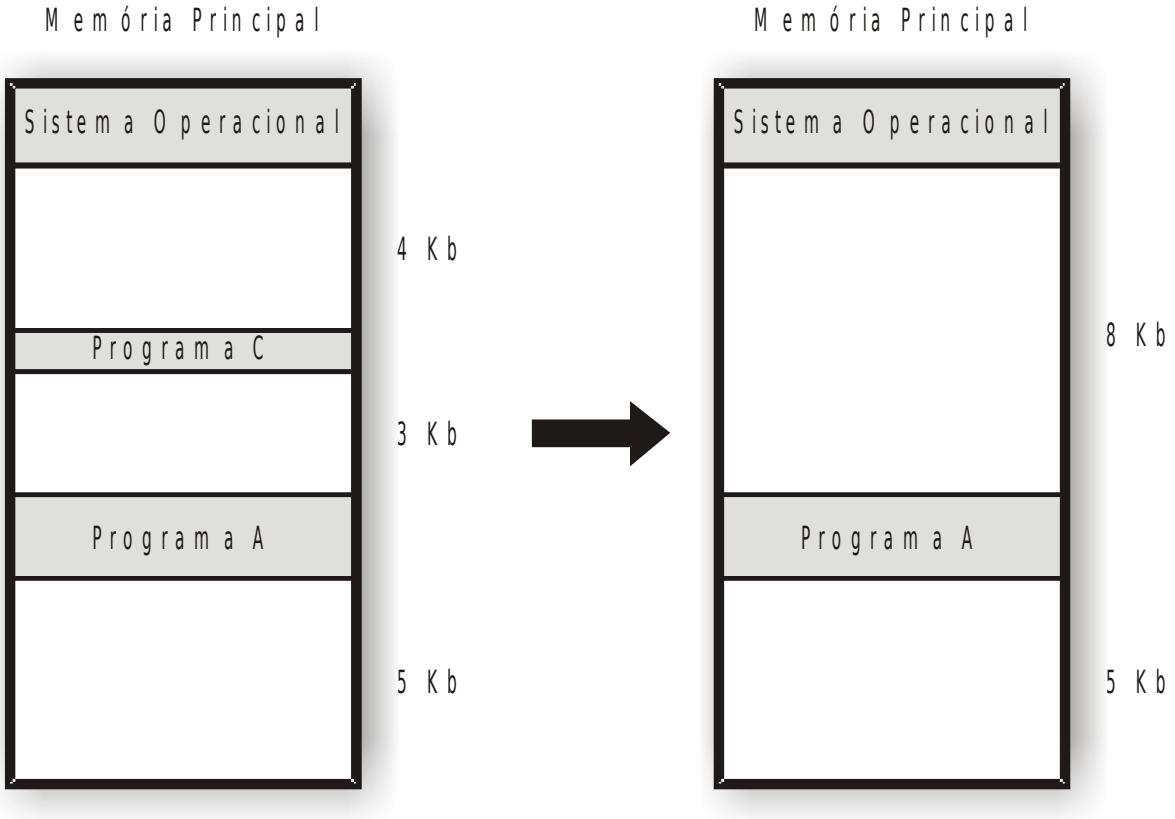
# Alocação Particionada Dinâmica

- Neste esquema também ocorre a fragmentação e ela aparecerá na medida em que os processos forem terminado e deixando espaços cada vez memores na memória, não permitindo o carregamento de outros processos.



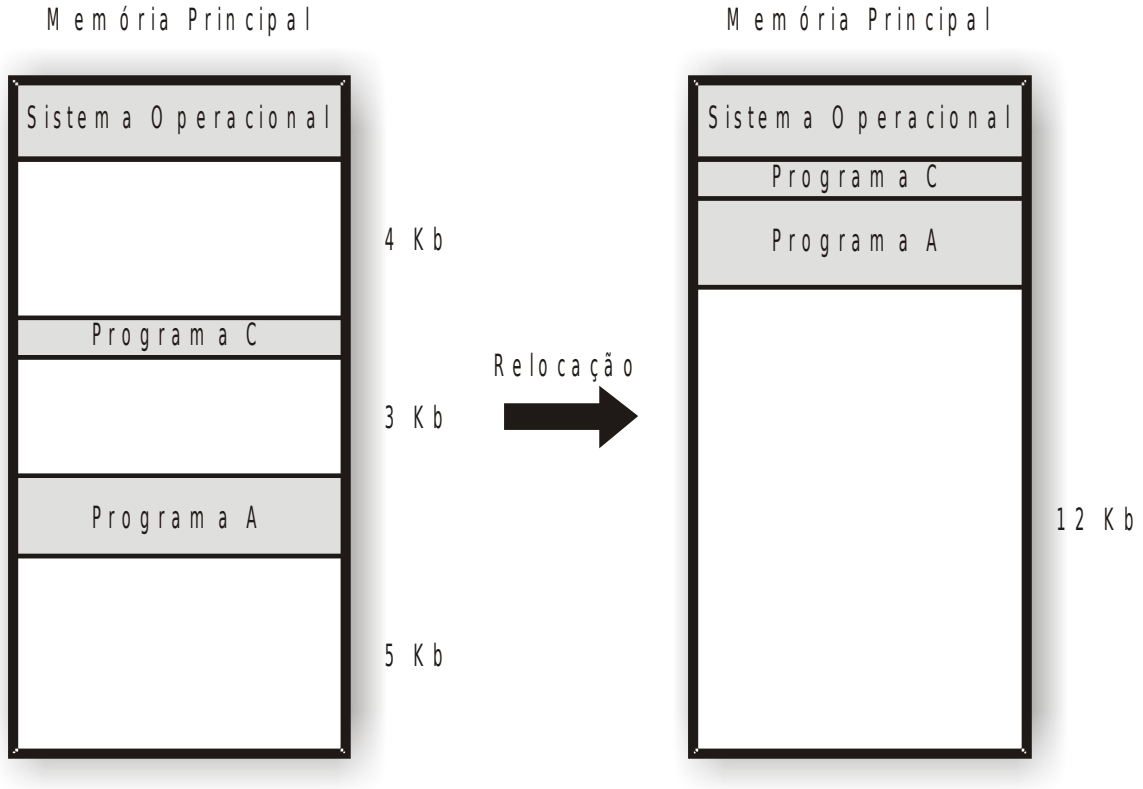
# Alocação Particionada Dinâmica

- Uma solução seria fazer com que os espaços adjacentes fossem reunidos, produzindo um único espaço de tamanho maior.



# Alocação Particionada Dinâmica

- A segunda maneira seria realizar uma relocação de todas as partições ocupadas, eliminando todos os espaços entre elas e criando-se uma única área livre contígua.



# Estratégias para Escolha da Partição

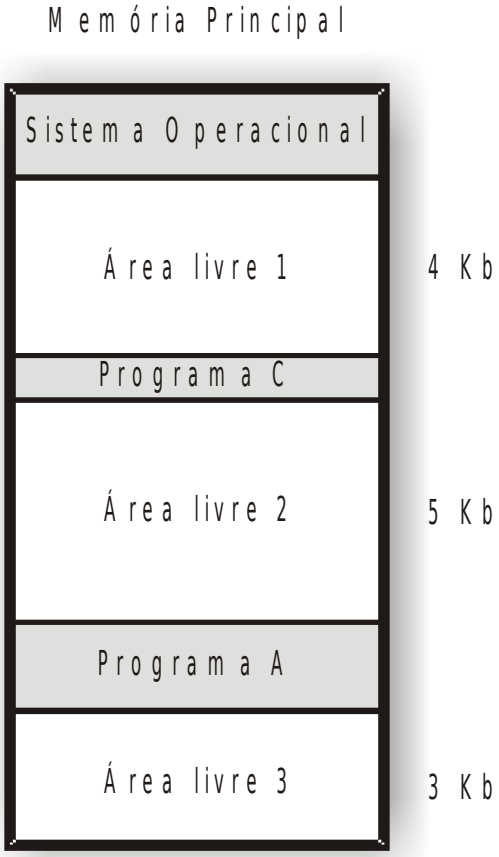
- Os sistemas operacionais implementam, basicamente, três estratégias para determinar em qual partição livre um programa será carregado para execução.

- Essas estratégias tentam evitar, ou diminuir, o problema da fragmentação antes que ela ocorra.

- Independente do algoritmo utilizado, o sistema possui uma lista de áreas livres ( free list ), com endereço de cada área livre e seu respectivo tamanho.

# Estratégias para Escolha da Partição

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



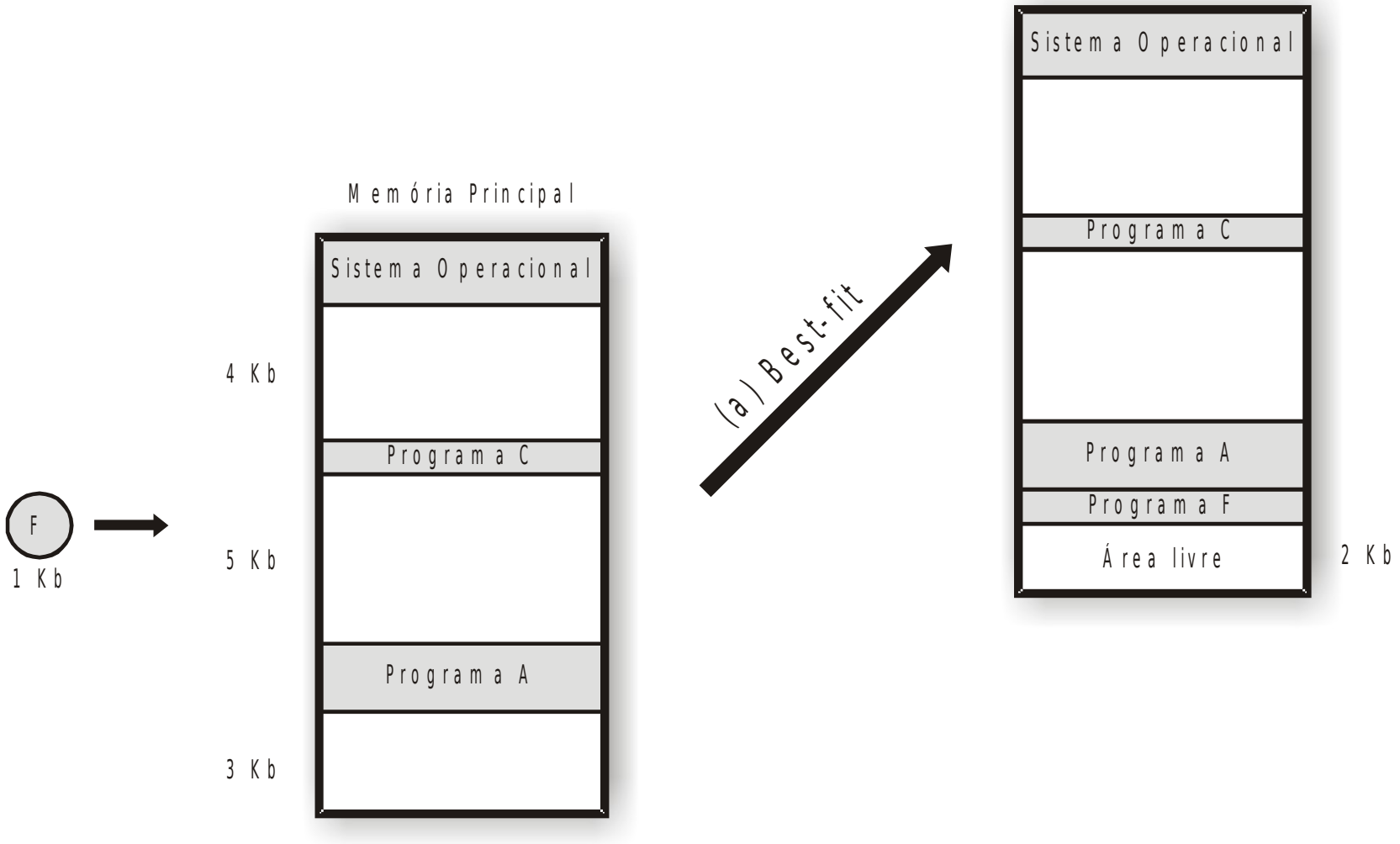
# Estratégias para Escolha da Partição

- BEST FIT
  - Esse mecanismo escolhe a melhor partição (best fit), ou seja, aquela em que o processo deixa o menor espaço sem utilização.

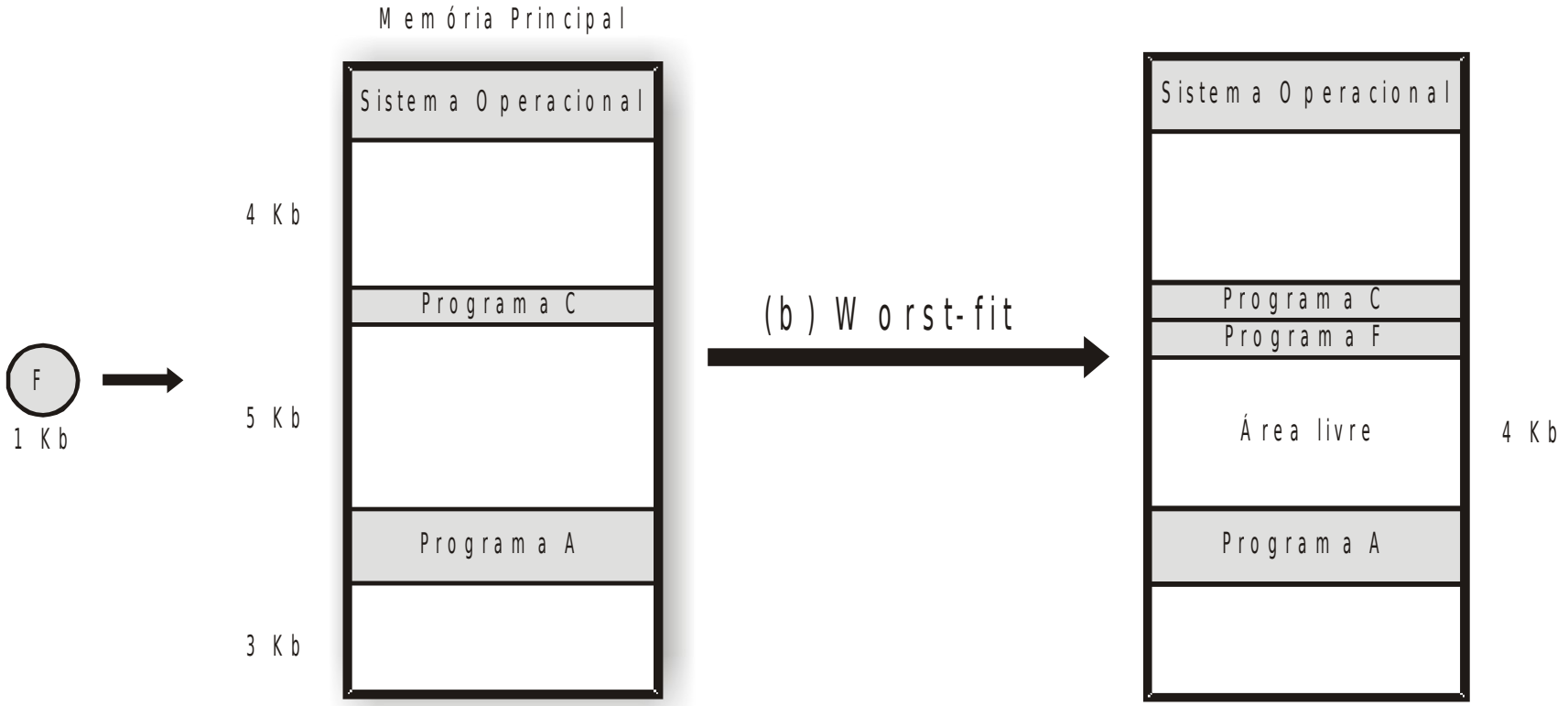
- WORST FIT
  - Esse mecanismo escolhe a pior partição (worst fit), ou seja, aquela em que o processo deixa o maior espaço sem utilização.

- FIRST FIT
  - Esse mecanismo escolhe a primeira partição (first fit) livre, de tamanho suficiente para carregar o processo.

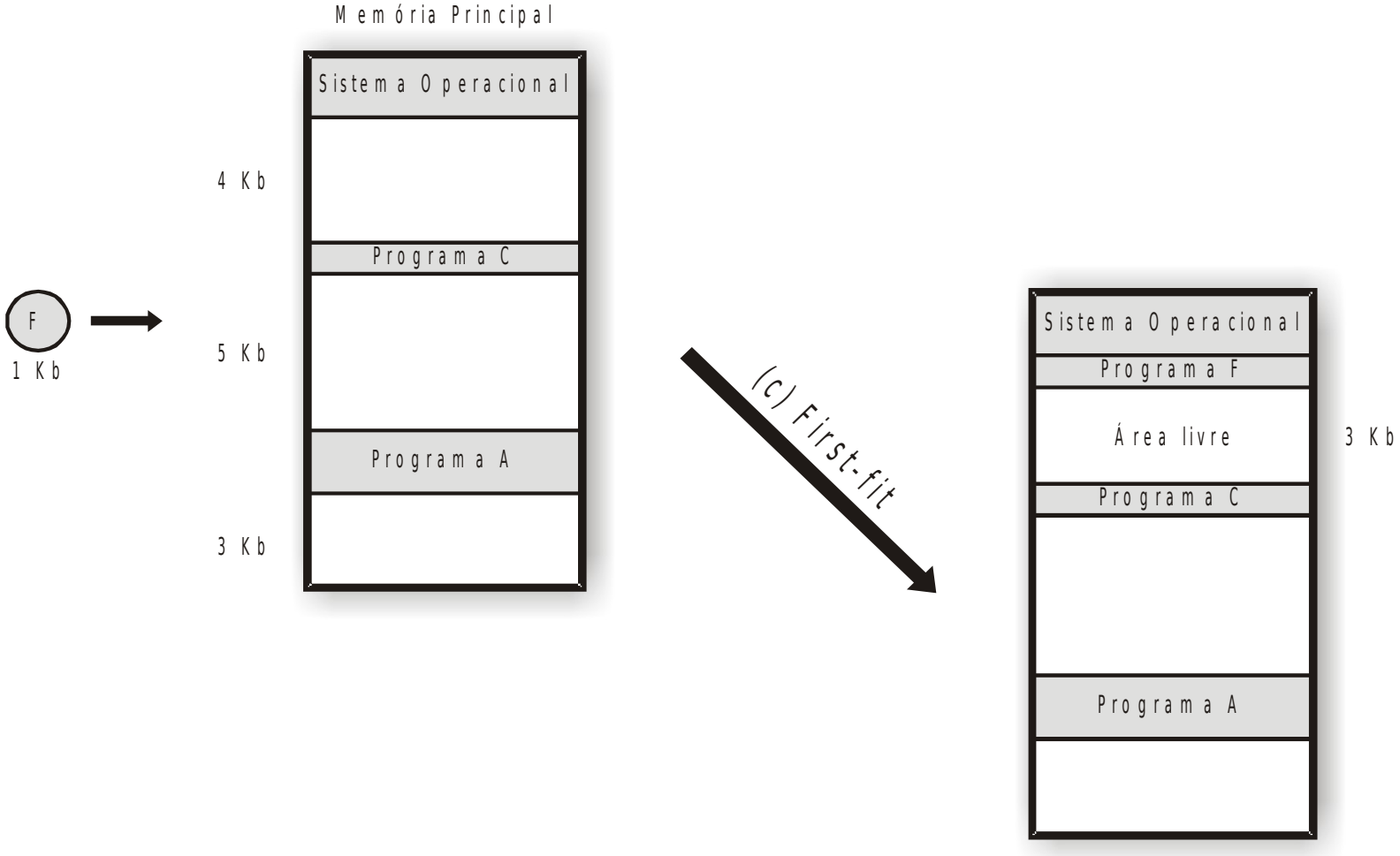
# Estratégias para Escolha da Partição



# Estratégias para Escolha da Partição



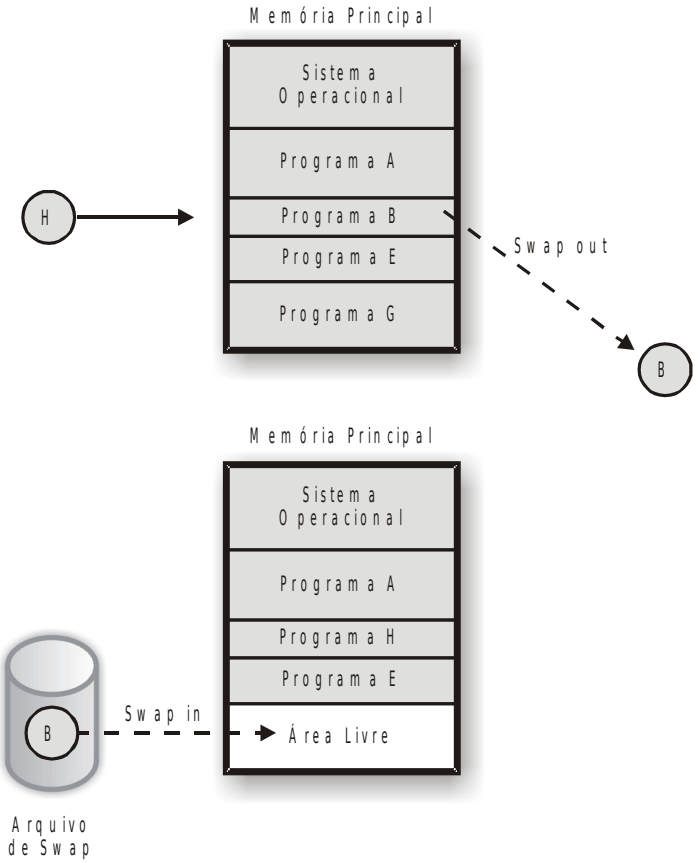
# Estratégias para Escolha da Partição



# Swapping

- A técnica de swapping veio para tentar resolver o problema de insuficiência de memória para todos os usuários.
- Nos esquemas apresentados até o momento, um processo permanecia na memória principal até o final de sua execução, inclusive quando realizava operações de entrada e saída.
- O swapping é uma técnica aplicada à gerência de memória, para processos que esperam por memória livre para serem processados.
- O sistema escolhe um processo residente que é levado da memória para o disco (swapped out), retornando posteriormente para a memória principal (swapped in), como se nada tivesse ocorrido.

# Swapping


Memória Principal
Sistema Operacional
Programa A
Programa B
Programa E
Programa G
H
Swap out
B
Memória Principal
Sistema Operacional
Programa A
Programa H
Programa E
Área Livre
Swap in
B
Arquivo de Swap

# Swapping

- Um dos problemas gerados pelo swapping é a relocação dos processos.
- O loader relocável permite que um processo seja colocado em qualquer posição de memória, porém a relocação é realizada no momento do carregamento.
- O conceito de swapping permitiu um maior compartilhamento de memória e, consequentemente, um maior throughput.
- Mostrou-se eficiente em ambientes onde existiam poucos usuários competindo pela memória e com aplicações pequenas.
- Seu maior problema é o custo das operações de entrada e saída.

# Gerenciamento de Memória

**Prof. Dr. José Luís Zem**
**Prof. Dr. Renato Kraide Soffner**
**Prof. Ms. Rossano Pablo Pinto**


**Faculdade de Tecnologia de Americana**
**Centro Paula Souza**