

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PIAUI

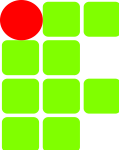
TADS

Sistemas Operacionais

Prof. Ricardo Ramos

Gerência de Memória

Capítulo 09

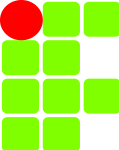


9.1 Introdução

Mesmo atualmente, com a redução de custo e conseqüente aumento da capacidade da memória principal, seu gerenciamento é um dos fatores mais importantes no projeto de SOs.

Monoprogramáveis - a gerência de memória não é muito complexa.

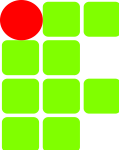
Multiprogramáveis - maximizar o número de usuários e aplicações utilizando eficientemente o espaço da MP.



9.2 Funções básicas

Mem. secundária → Mem. principal → executados

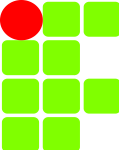
Como o tempo de acesso à mem. secundária (MS) é muito superior ao tempo de acesso à mem. principal, o SO deve buscar reduzir o número de operações de E/S à mem. secundária, senão sérios problemas de desempenho podem ocorrer.



9.2 Funções básicas

Objetivo: manter na MP o maior número de processos residentes, permitindo maximizar o compartilhamento do processador e demais recursos computacionais.

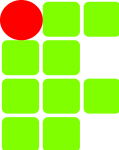
Swapping - transferência temporária de processos residentes na MP para a MS.



9.2 Funções básicas

Outra preocupação na gerência de memória é permitir a execução de programas que sejam maiores que a memória física disponível, implementada através de técnicas como o overlay e a memória virtual.

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



9.3 Alocação contígua simples

Implementada nos primeiros SOs, porém ainda está presente em alguns sistemas monoprogramáveis.

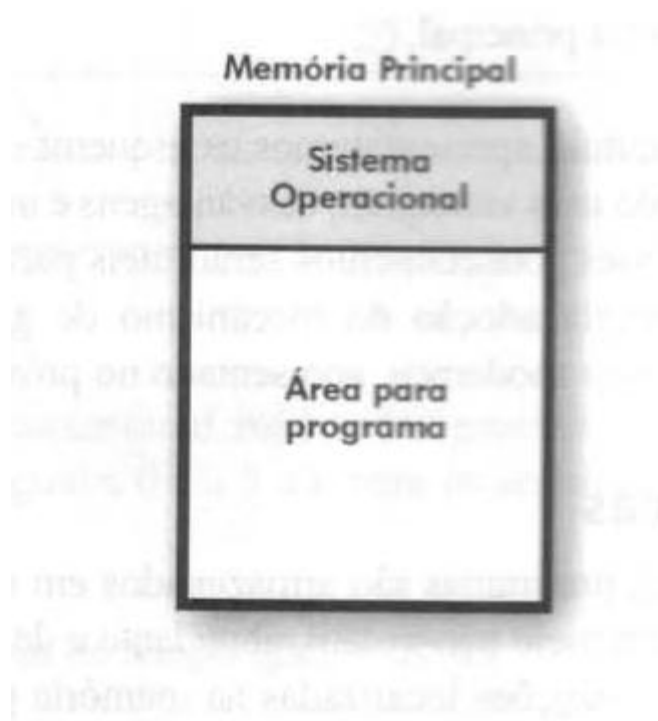
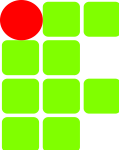


Fig. 9.1 Alocação contígua simples.



9.3 Alocação contígua simples

Nesse esquema, o usuário tem controle sobre toda a MP, podendo ter acesso a qualquer posição de memória, inclusive a área do SO.

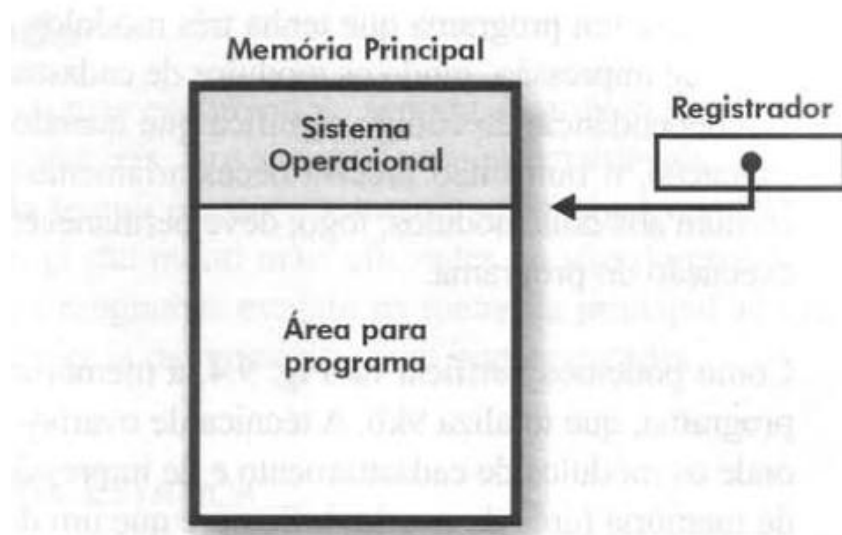
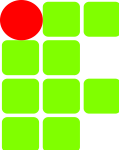
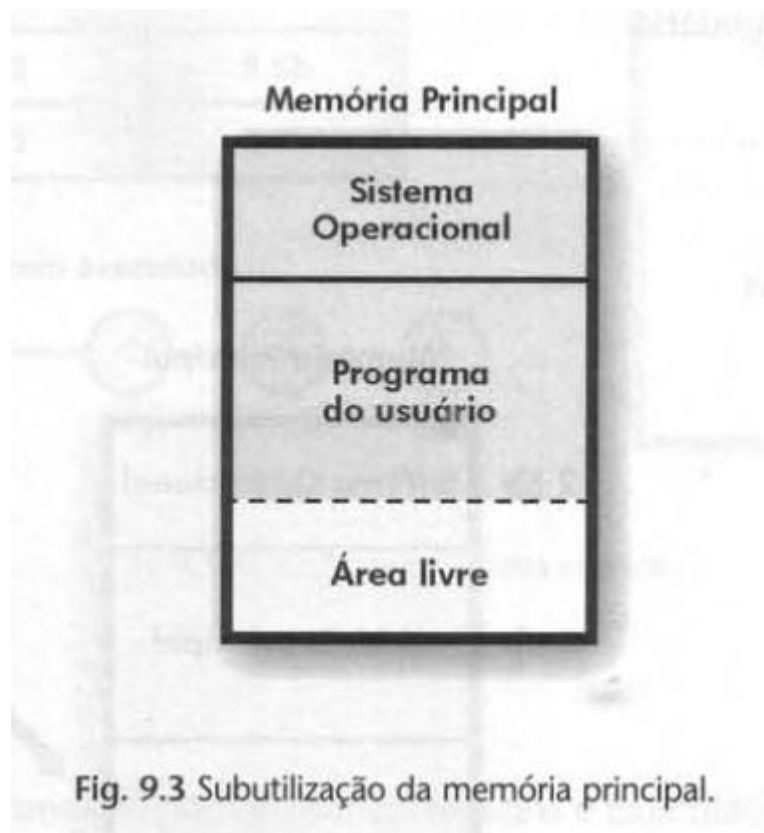


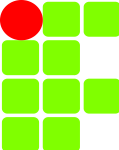
Fig. 9.2 Proteção na alocação contígua simples.



9.3 Alocação contígua simples

Não permite a utilização eficiente dos recursos computacionais.

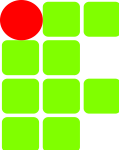




9.4 Técnica de Overlay

Na alocação contígua simples todos os programas estão limitados ao tamanho da área de MP disponível para o usuário.

A técnica de overlay divide o programa em módulos, de forma que seja possível a execução independente de cada módulo, utilizando uma mesma área de memória.



9.4 Técnica de Overlay

O módulo principal deve permanecer na memória o tempo todo.

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

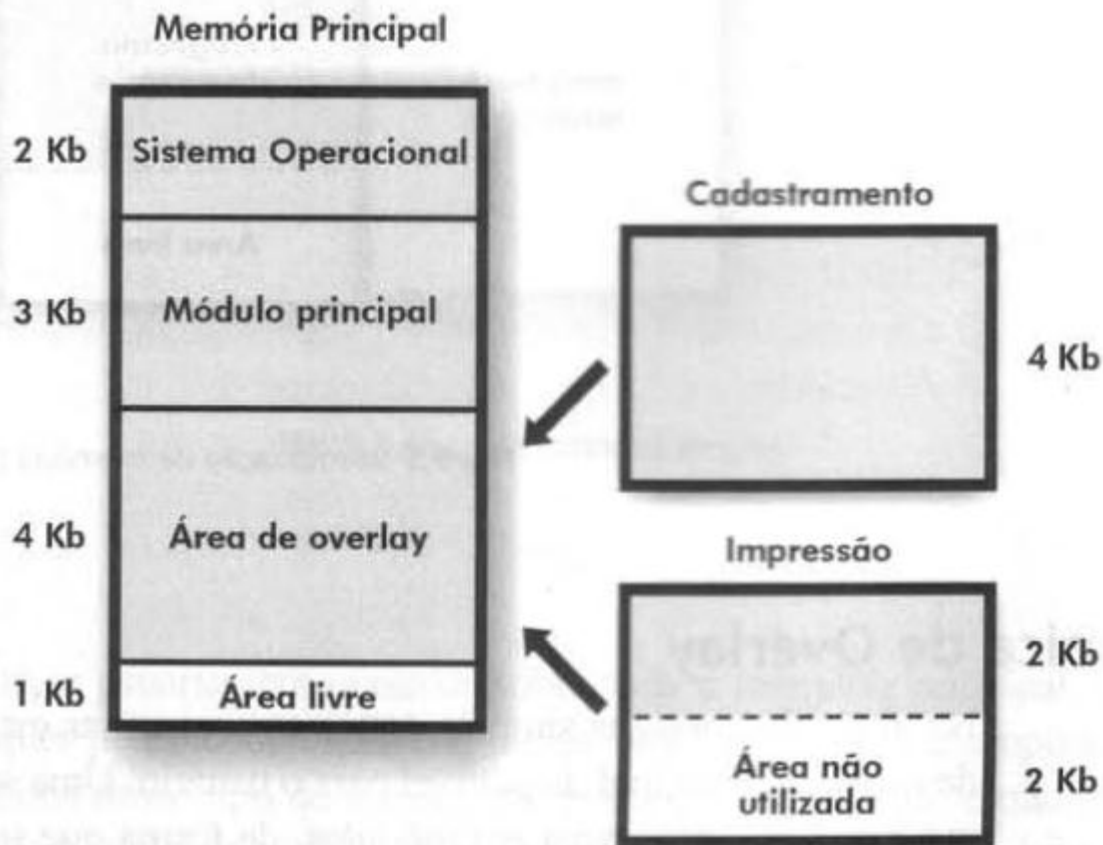
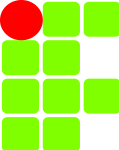


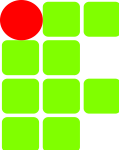
Fig. 9.4 Técnica de overlay.



9.5 Alocação particionada

9.5.1 - Alocação particionada estática

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



9.5 Alocação particionada

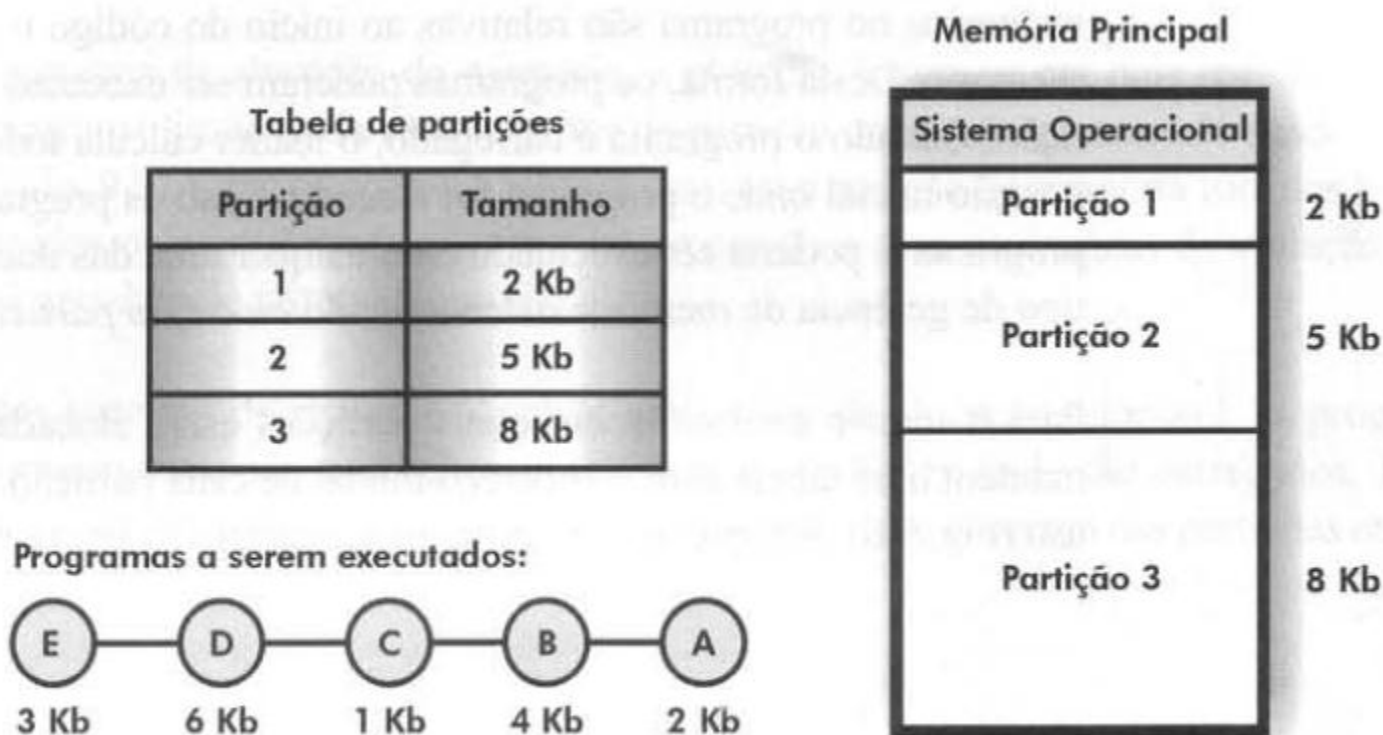
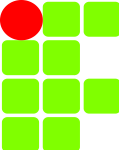


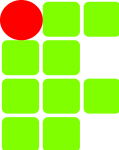
Fig. 9.5 Alocação particionada estática.



9.5 Alocação particionada

9.5.1 - Alocação particionada estática

Inicialmente, os programas só podiam ser carregados e executados em apenas uma partição específica, mesmo se outras estivessem disponíveis.



9.5 Alocação particionada

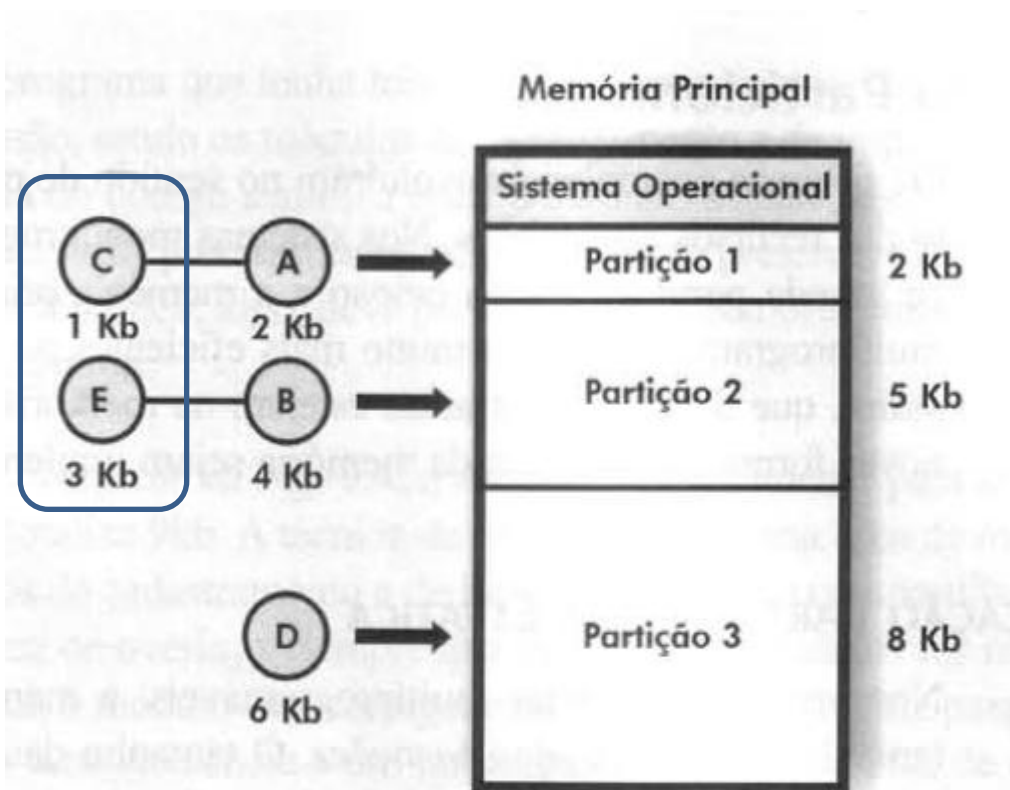
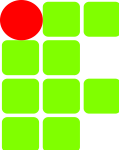


Fig. 9.6 Alocação particionada estática absoluta.

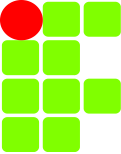


9.5 Alocação particionada

9.5.1 - Alocação particionada estática

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

Dessa forma, os programas puderam ser executados a partir de qualquer partição.



9.5 Alocação particionada

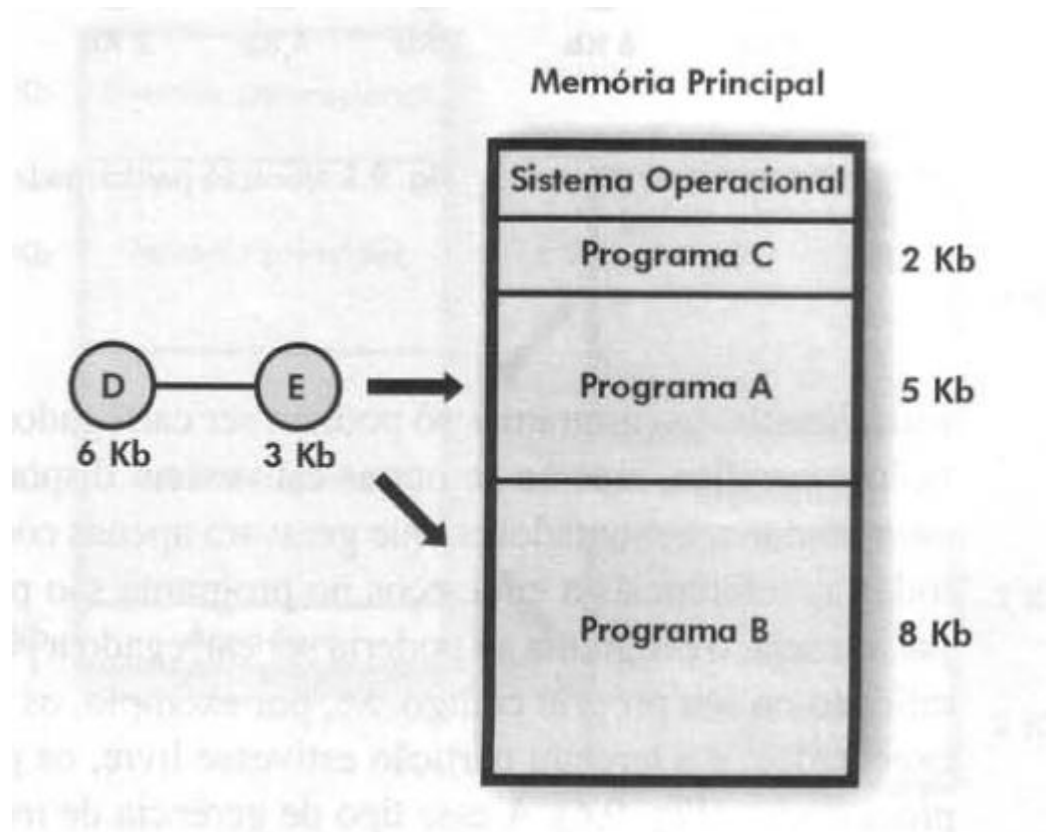
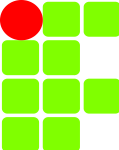


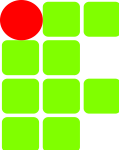
Fig. 9.7 Alocação particionada estática relocável.



9.5 Alocação particionada

9.5.1 - Alocação particionada estática

Para manter o controle sobre quais partições estão alocadas, a gerência de memória mantém uma tabela com o endereço inicial de cada partição, seu tamanho e se está em uso.



9.5 Alocação particionada

9.5.1 - Alocação particionada estática

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

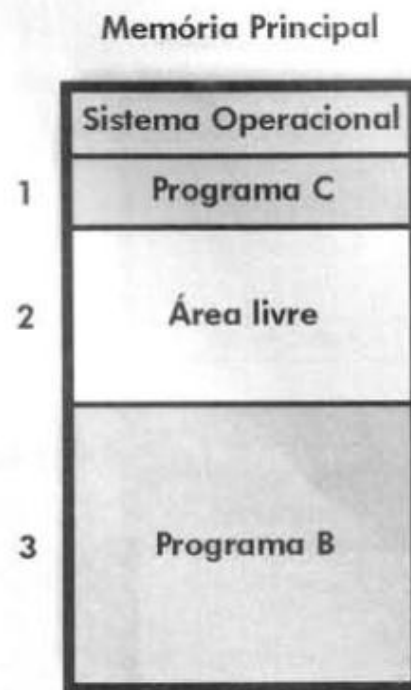
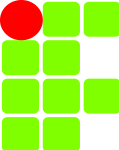


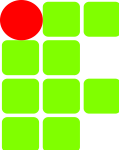
Fig. 9.8 Tabela de alocação de partições.



9.5 Alocação particionada

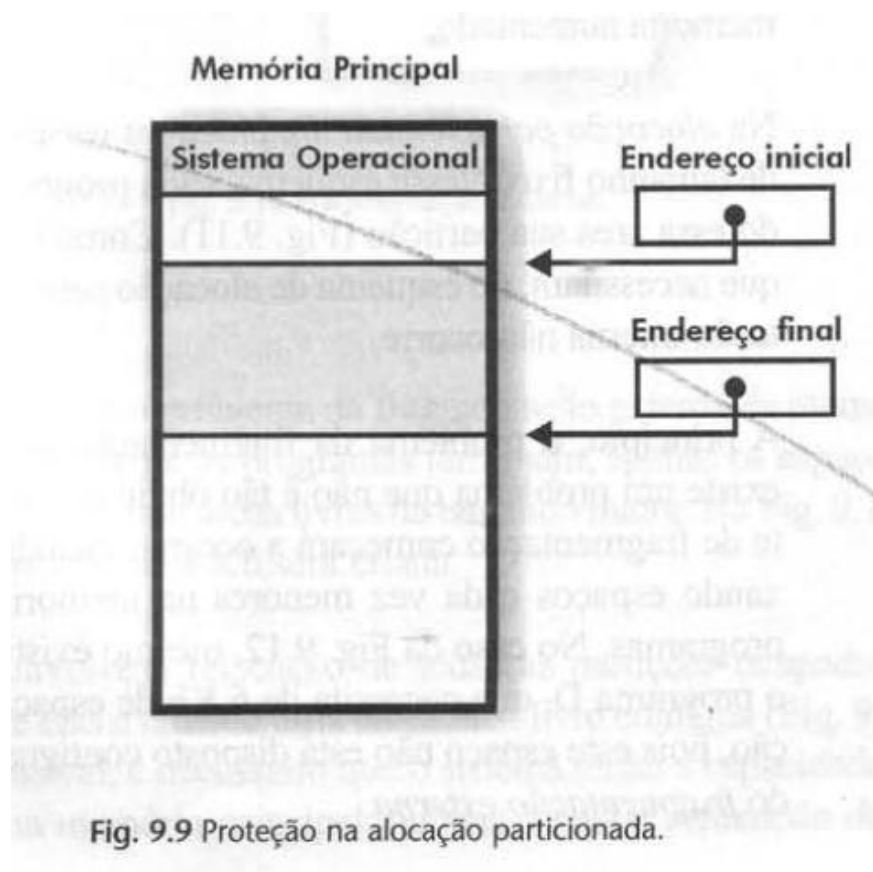
9.5.1 - Alocação particionada estática

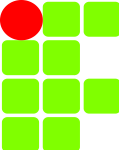
Nesse esquema de alocação de memória a proteção baseia-se em dois registradores, que indicam os limites inferior e superior da partição onde o programa está sendo executado.



9.5 Alocação particionada

9.5.1 - Alocação particionada estática

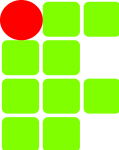




9.5 Alocação particionada

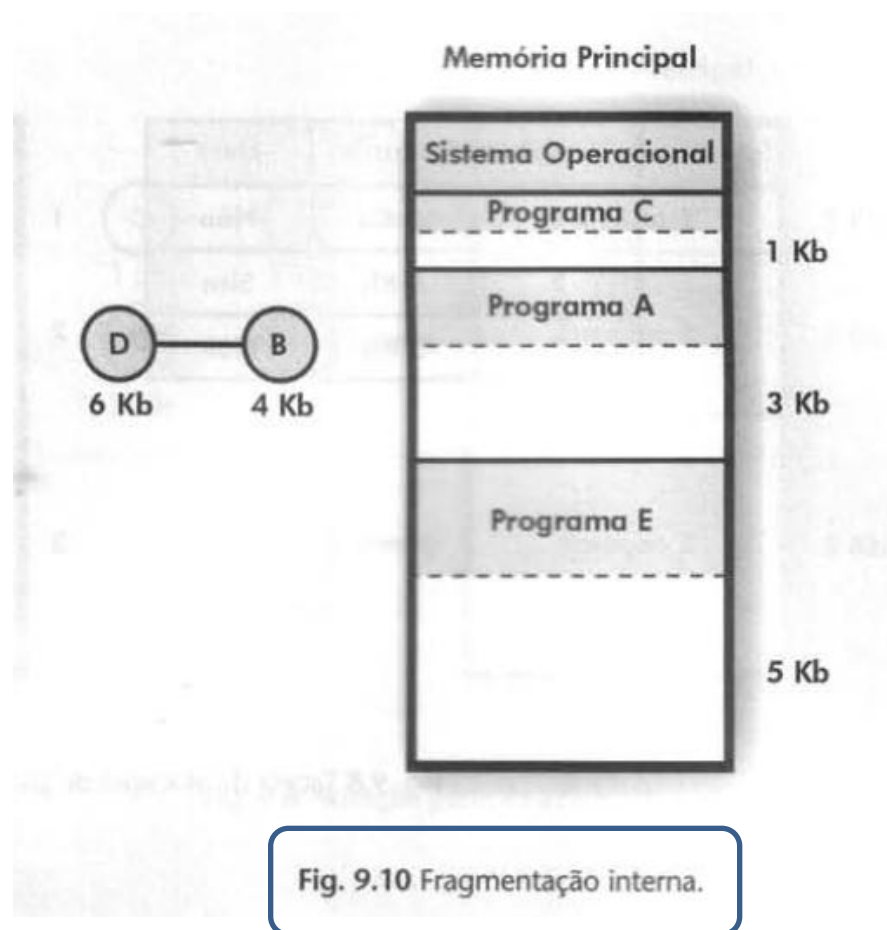
9.5.1 - Alocação particionada estática

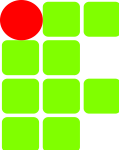
Tanto nos sistemas de alocação absoluta quanto nos de alocação relocável os programas, normalmente não preenchem totalmente as partições onde são carregados.



9.5 Alocação particionada

9.5.1 - Alocação particionada estática





9.5 Alocação particionada

9.5.2 - Alocação particionada dinâmica (ou variável)

Resolver o problema da fragmentação interna e aumentar o grau de compartilhamento da memória.

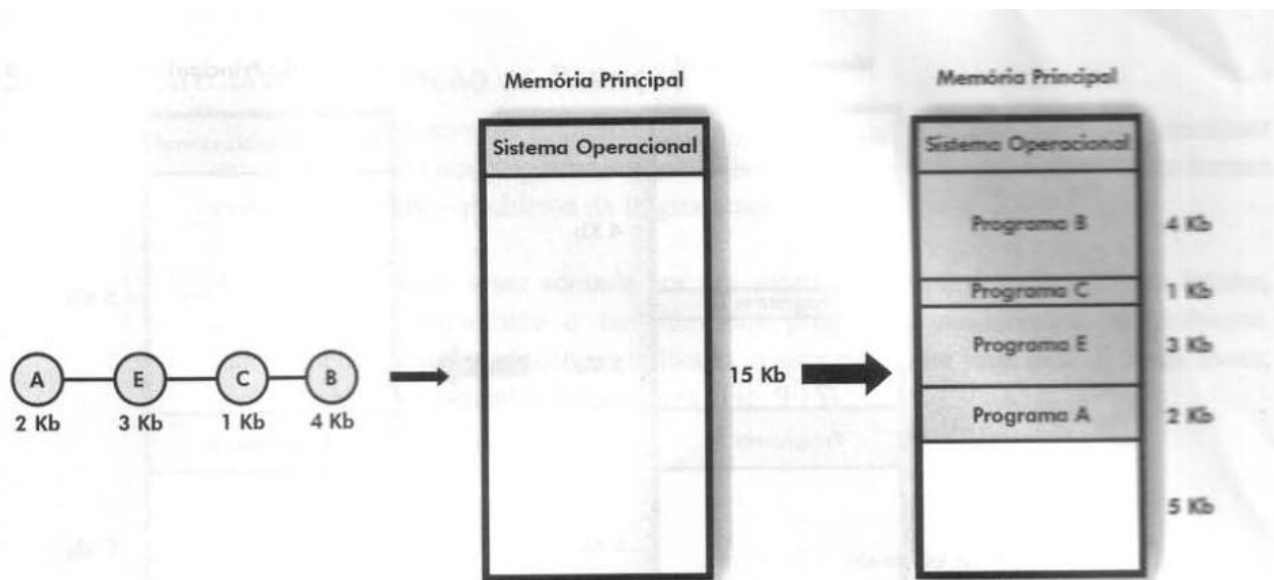
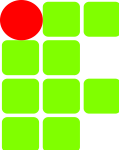


Fig. 9.11 Alocação particionada dinâmica.



9.5 Alocação particionada

9.5.2 - Alocação particionada dinâmica (ou variável)

Problema: Fragmentação externa

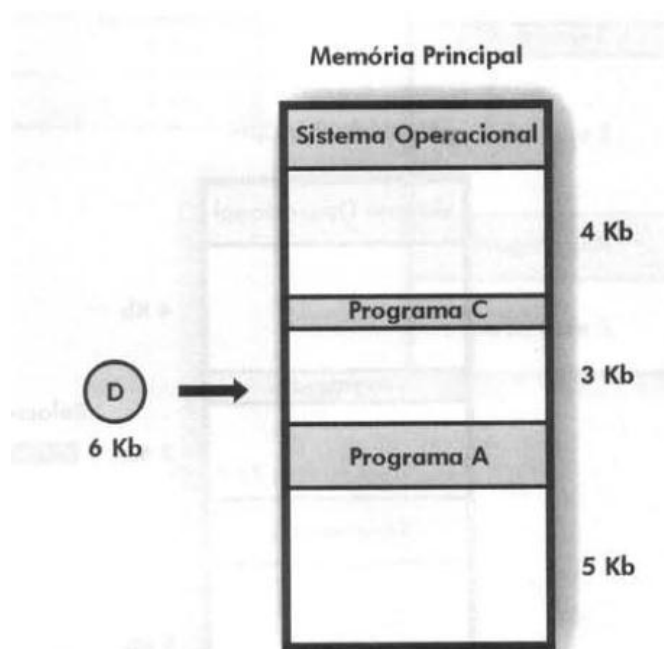
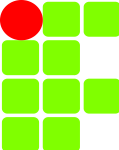


Fig. 9.12 Fragmentação externa.



9.5 Alocação particionada

9.5.2 - Alocação particionada dinâmica (ou variável)

Soluções para o problema da fragmentação externa.

1ª Solução

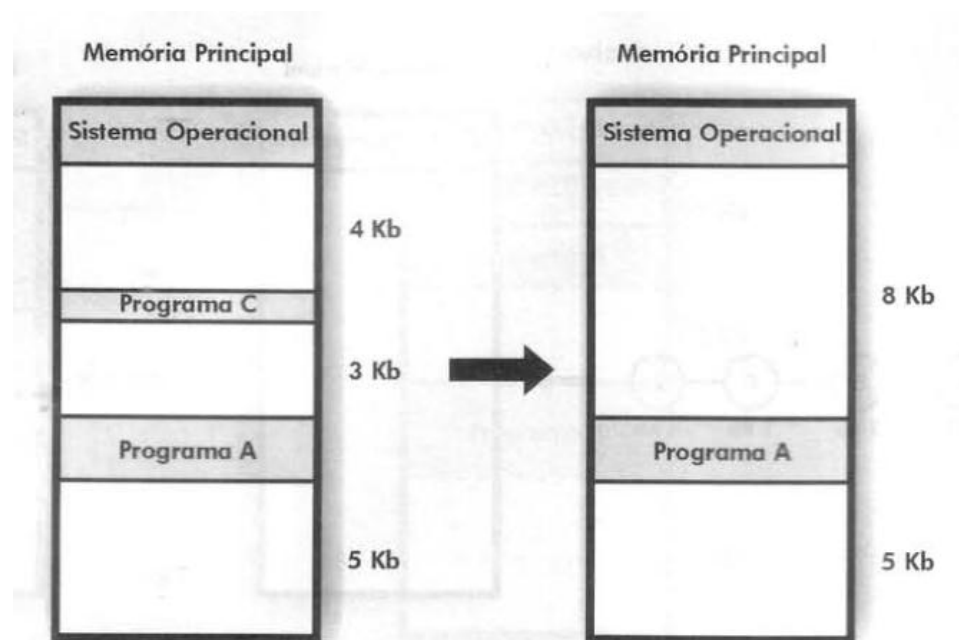
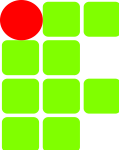


Fig. 9.13 Solução para a fragmentação externa (a).



9.5 Alocação particionada

9.5.2 - Alocação particionada dinâmica (ou variável)

Soluções para o problema da fragmentação externa.

2ª Solução – Relocação dinâmica

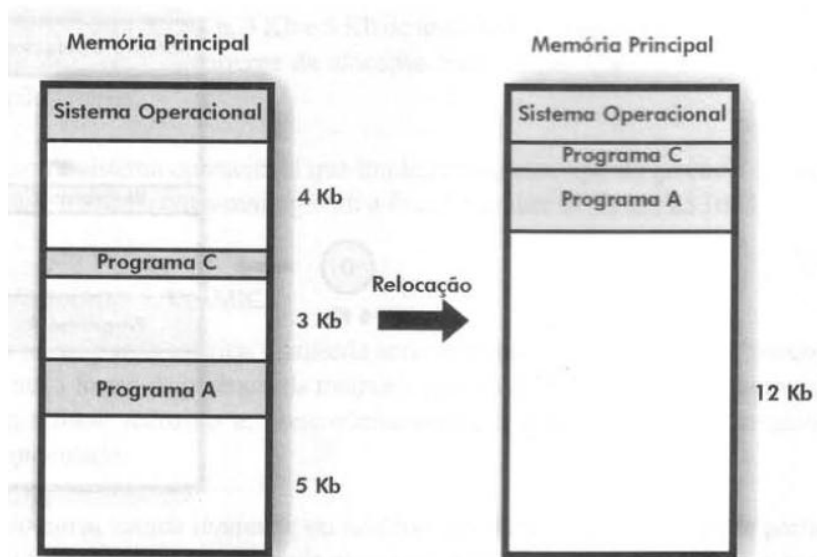
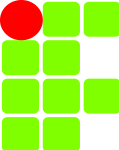


Fig. 9.14 Solução para a fragmentação externa (b).



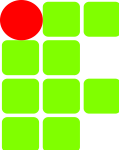
9.5 Alocação particionada

9.5.2 - Alocação particionada dinâmica (ou variável)

O mecanismo anterior de compactação, é também conhecido como *alocação particionada dinâmica com relocação*.

Vantagem: reduz em muito o problema da fragmentação.

Desvantagem: complexidade do seu algoritmo, o consumo de recursos do sistema, como processador e área em disco podem torná-lo inviável.



9.5 Alocação particionada

9.5.3- Estratégias de Alocação de Partição

São três e tentam evitar ou diminuir o problema da fragmentação externa.

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

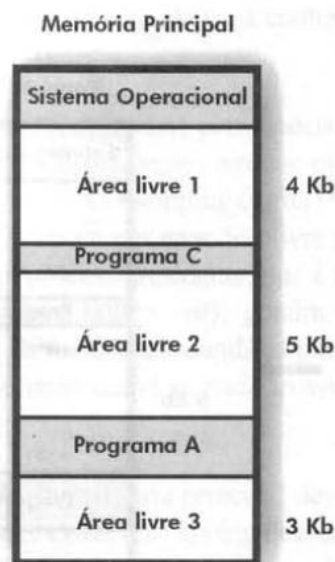
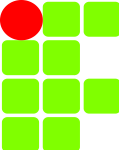


Fig. 9.15 Lista de áreas livres.



9.5 Alocação particionada

9.5.3- Estratégias de Alocação de Partição

1. Best-Fit:

(deixa a menor área livre)

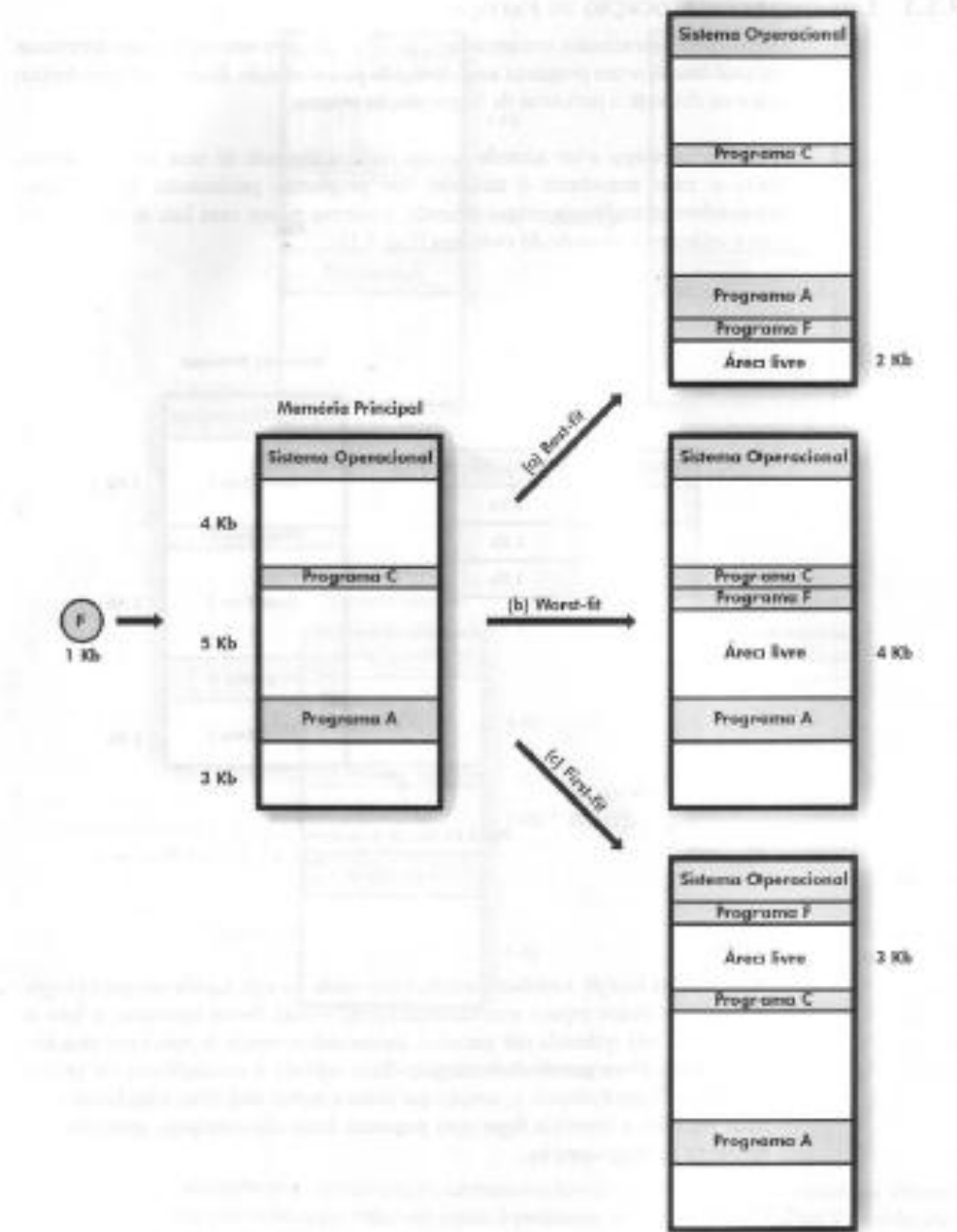
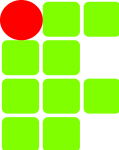
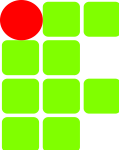


Fig. 9.16 Estratégias para a escolha da partição.



9.5 Alocação particionada

9.5.3- Estratégias de Alocação de Partição

1. Worst-Fit:

(deixa a maior área livre)

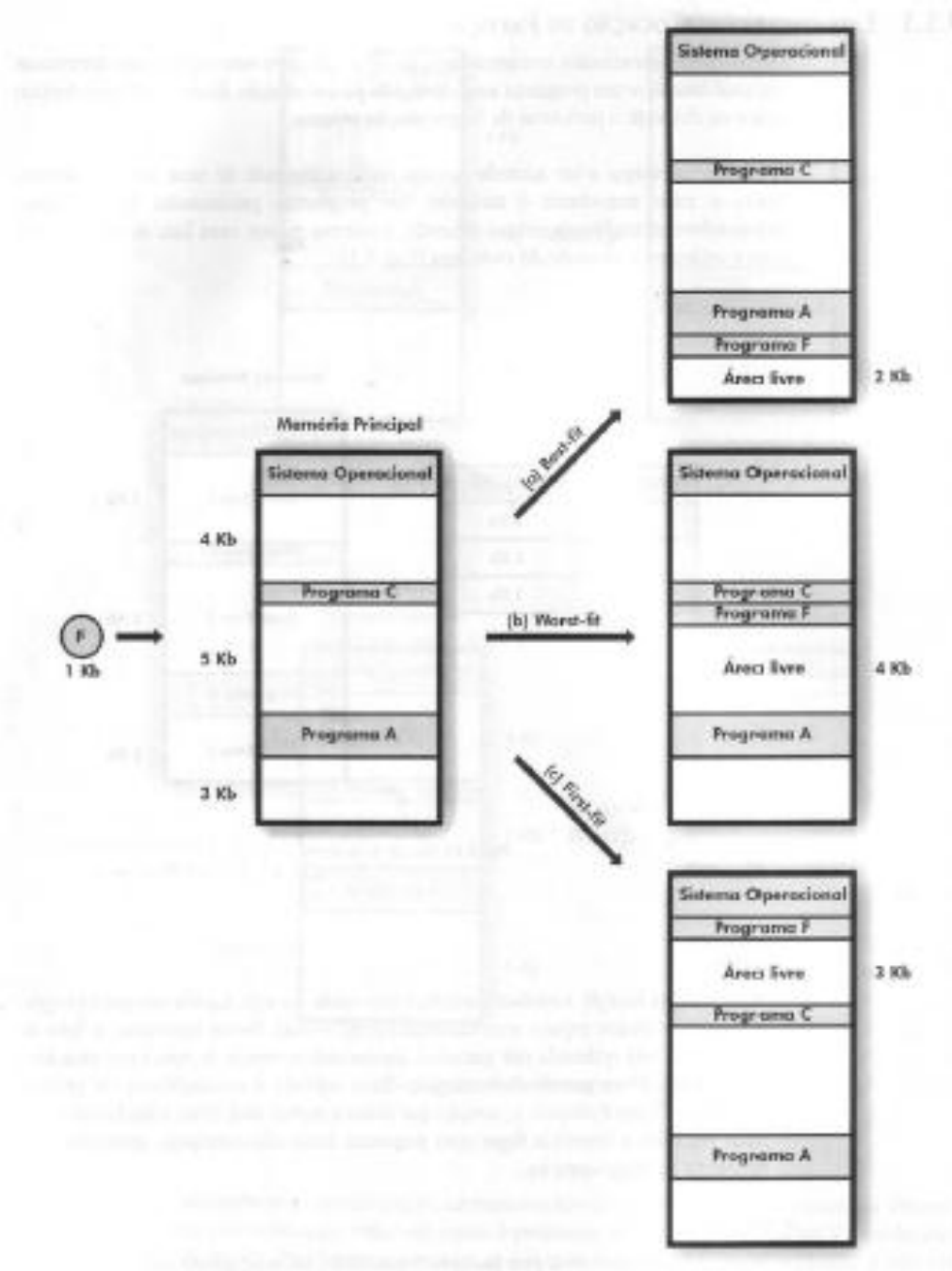
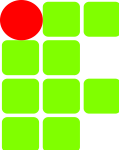
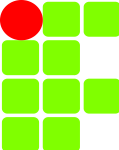


Fig. 9.16 Estratégias para a escolha da partição.



9.5 Alocação particionada

9.5.3- Estratégias de Alocação de Partição

1. First-Fit:

(a primeira partição livre de tamanho suficiente para carregar o programa é escolhida)

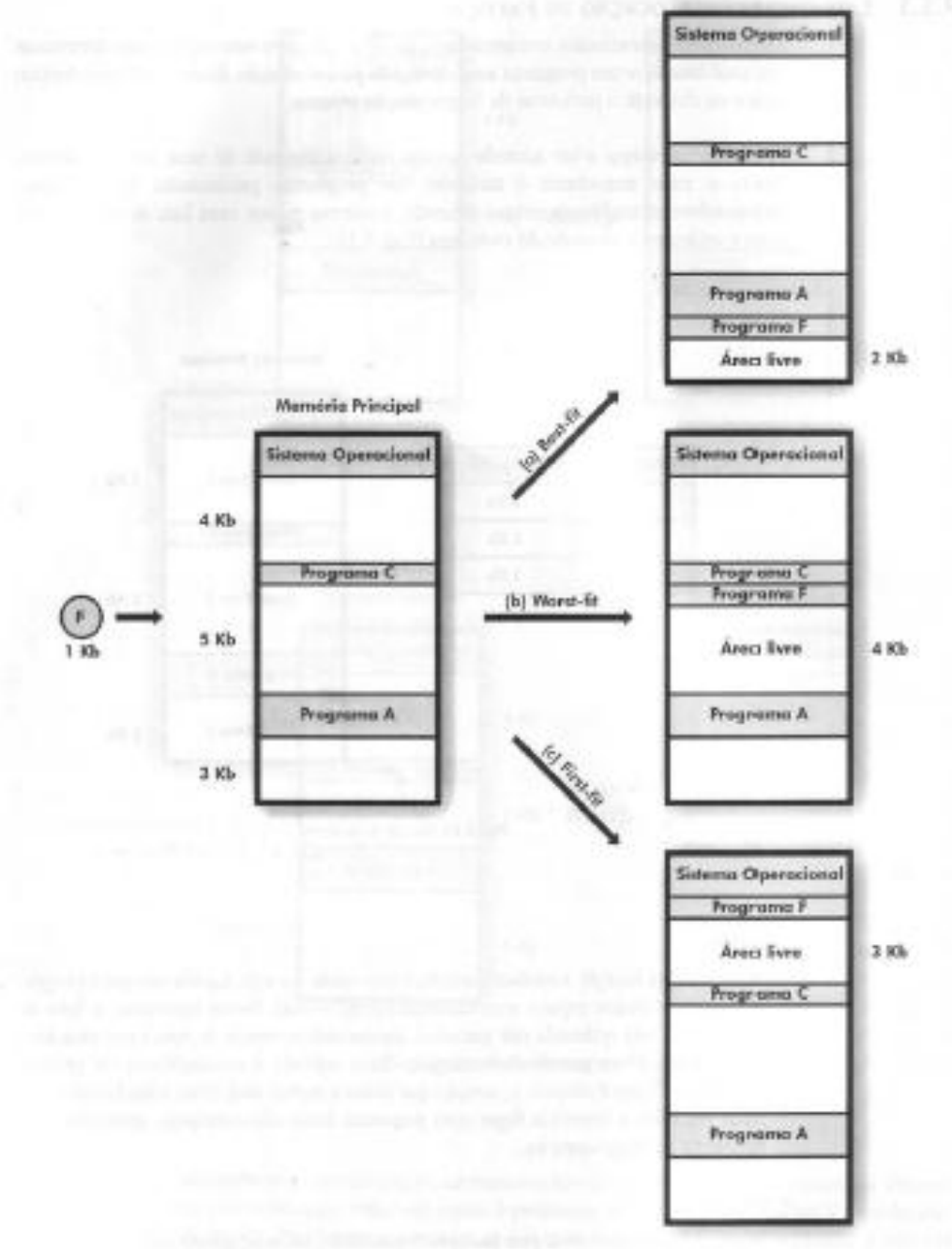
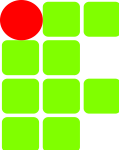
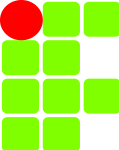


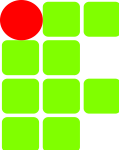
Fig. 9.16 Estratégias para a escolha da partição.



9.6 Swapping

Introduzida para contornar o problema da insuficiência de memória principal.

O swapping é uma técnica aplicada à gerência de memória para programas que esperam por memória livre para serem executados.



9.6 Swapping

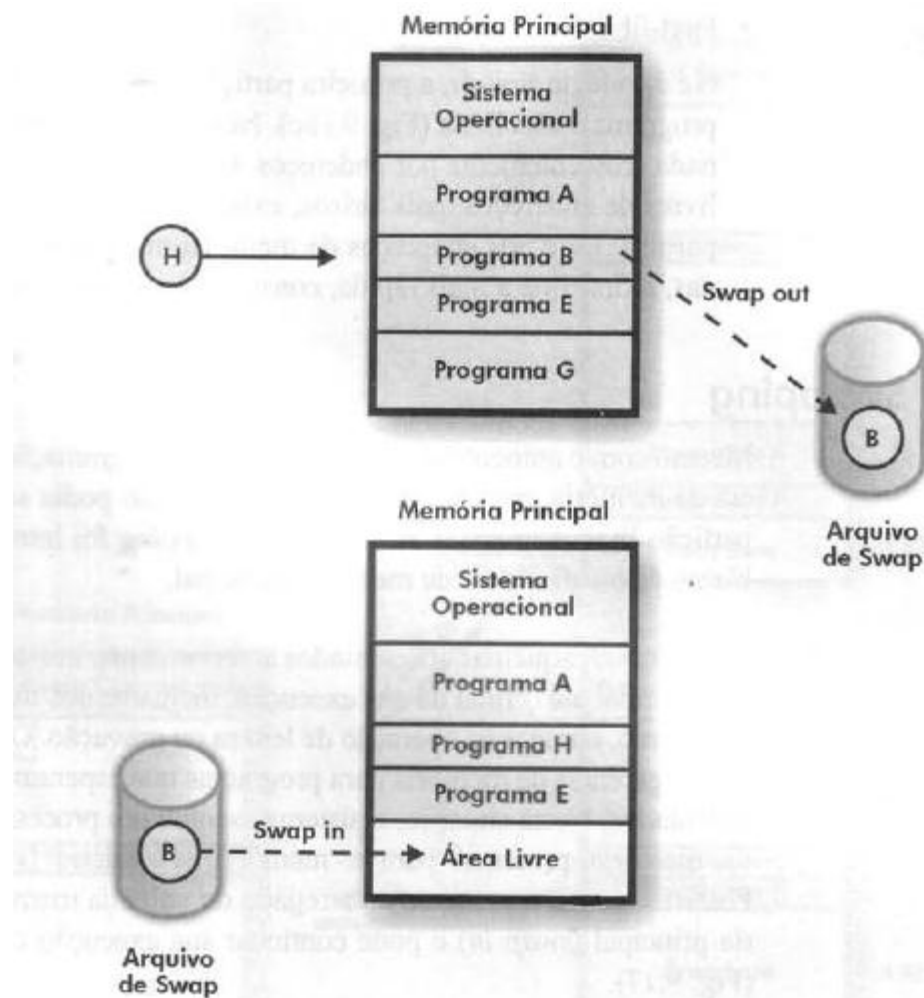
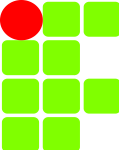


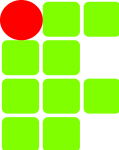
Fig. 9.17 Swapping.



9.6 Swapping

O algoritmo de escolha do processo a ser retirado da memória principal deve priorizar aquele com menores chances de ser escalonado para evitar o swapping desnecessário de um processo que será executado logo em seguida.

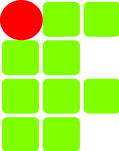
Para que a técnica de swapping seja implementada é essencial que o sistema ofereça um loader que implemente a *relocação dinâmica* de programas.



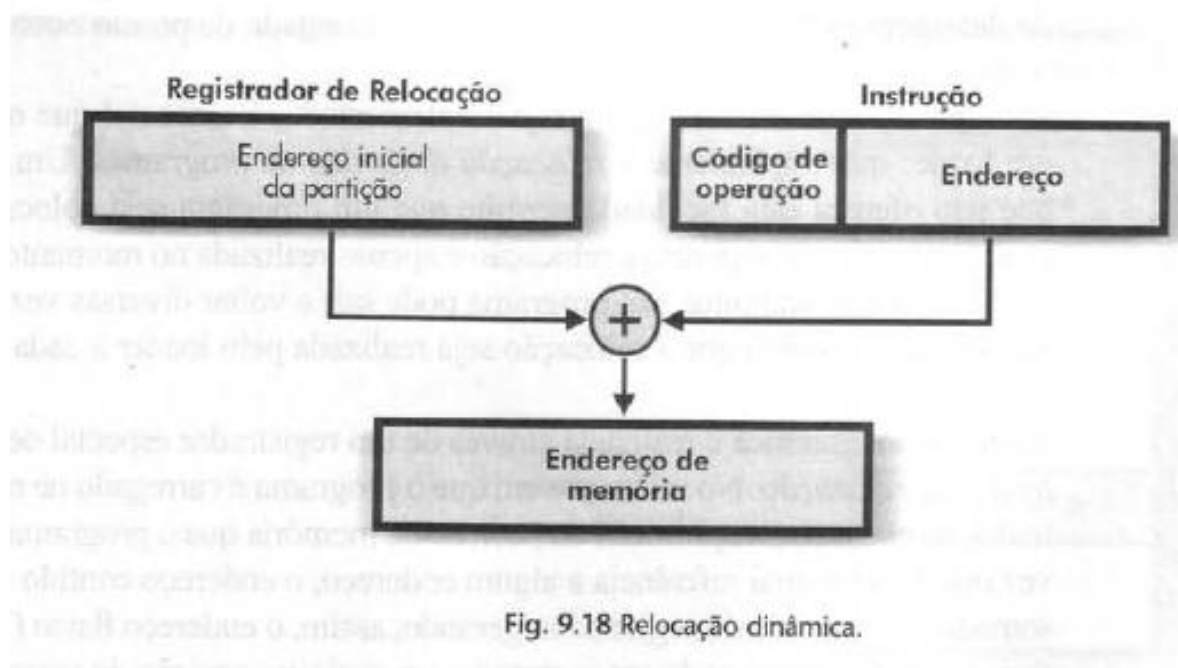
9.6 Swapping

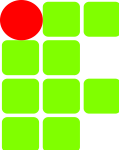
O algoritmo de escolha do processo a ser retirado da memória principal deve priorizar aquele com menores chances de ser escalonado para evitar o swapping desnecessário de um processo que será executado logo em seguida.

Para que a técnica de swapping seja implementada é essencial que o sistema ofereça um loader que implemente a *relocação dinâmica* de programas.



9.6 Swapping





9.6 Swapping

Vantagens: maior compartilhamento da MP e maior utilização dos recursos do sistema computacional

Problema: elevado custo das operações de E/S (swap in/out) e *thrashing*.