Arquitetura de Sistemas Operacionais

Francis Berenger Machado
Luiz Paulo Maia

Capítulo 9
Gerência de Memória

Sumário

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

Introdução

- Enquanto nos sistemas monoprogramáveis a gerência da memória não é muito complexa, nos sistemas multiprogramáveis essa gerência se torna crítica, devido à necessidade de se maximizar o número de usuários e aplicações utilizando eficientemente o espaço da memória principal.

Funções Básicas

- Manter o maior número de processos na memória
- Maximizar o compartilhamento da UCP e demais recursos
- Swapping
- Execução de programas maiores que memória disponível
- Proteção
- Compartilhamento

Esquema de uso

submete **JOB** para execução

MEMÓRIA PRINCIPAL

Sistema Operacional

LOCATOR
procura na biblioteca

MEMÓRIA SECUNDÁRIA

Biblioteca de programas

LOADER
carrega programa na memória

Código do Programa e Recursos

SWAPPING

PAGEFILE

Quando o espaço de memória é insuficiente

EXECUÇÃO

Alocações de Memória

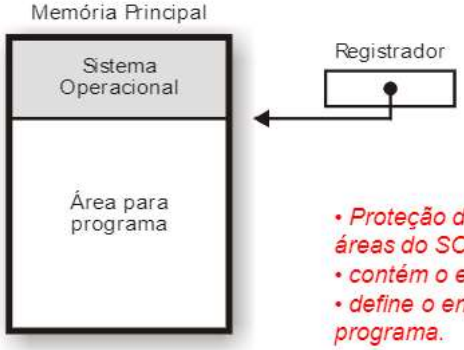
- **ALOCAÇÃO DE MEMÓRIA:**
  - **Contígua**
    - **Simples**
    - **Particionada Estática ou Fixa**
      - **Absoluta**
      - **Relocável**
    - **Particionada Dinâmica**
  - **Não-Contígua ou Memória Virtual**
    - **Paginação**
    - **Segmentação**
    - **Segmentação com Paginação**

Alocação Contígua Simples

- **implementado nos primeiros SO**
- *monoprogramável*
- *memória contígua – contínua, adjacentes*
- *dividida em duas áreas:*
  - *para Sistema Operacional*
  - *para programas do usuário*
- *não pode exceder o espaço disponível*
- *usuário tem controle total sobre o uso da memória*

Alocação Contígua Simples

- Proteção



- *Proteção do SO, o registrador delimita as áreas do SO*
- *contém o endereço final do SO*
- *define o endereçamento da área do programa.*

Alocação Contígua Simples

- Subutilização da memória

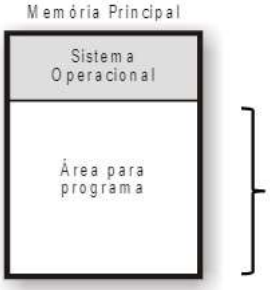
- *memória subutilizada*
- *utilização ineficiente*
- *monousuário – monoprogramável*
- *desperdício de recurso caro*

Memória alocada para o programa

Resto da memória não utilizada "Fragmentação Interna"

Alocação Contígua Simples

- o programador deve desenvolver suas aplicações, preocupado, apenas, em **não ultrapassar o espaço de memória disponível**, ou seja, a diferença entre o tamanho total da memória principal e área ocupada pelo Sistema Operacional.



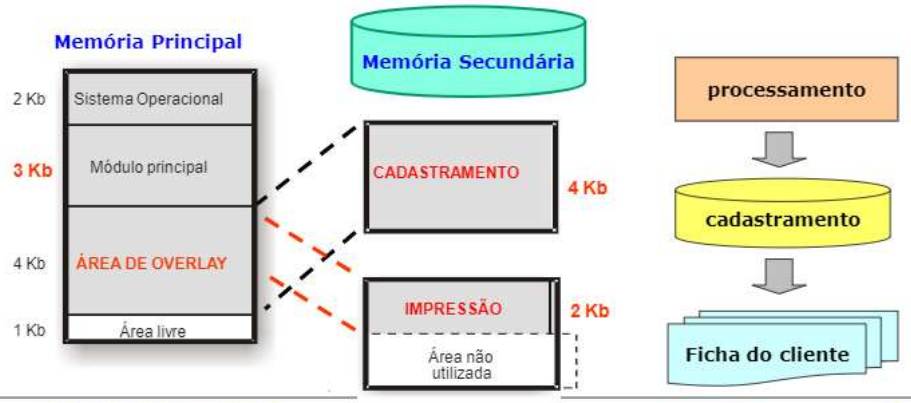
- Espaço de Memória Principal disponível para programas do usuário
- Não pode exceder o tamanho

Técnica de Overlay

- Na alocação contígua simples, todos os programas **estão limitados** ao tamanho da área de memória principal disponível para o usuário.
- Uma solução encontrada para o problema é **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

- **A memória de 8 Kb é insuficiente para o Programa com 9 kb**
- **dividir em 3 módulos (3 kb, 4 kb e 2 Kb)**
- **módulo principal residente na memória de 3 kb**
- **Os Módulos Overlays compartilham a Área de Overlay de 4 kb**
- **Área de Overlay = 4 kb, definido pelo maior módulo (Ex. Cadastramento)**
- **Carregado da Memória Secundária para a Area de Overlay qdo. referenciado.**



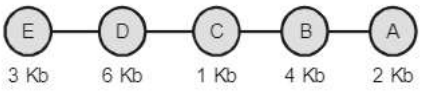
Alocação Contígua Particionada Estática

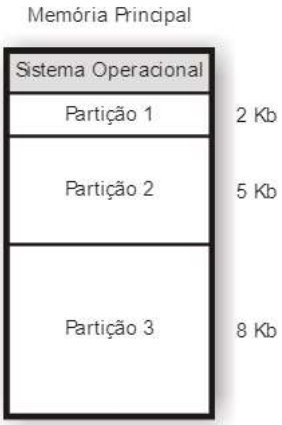
- *Partições são fixas e definidas na inicialização do S.O.*
- *Tamanhos são determinados baseados nos tamanhos dos programas e da sua necessidade de execução.*
- *Pode-se mudar os tamanhos das partições reinicializando o SO.*

Tabela de partições

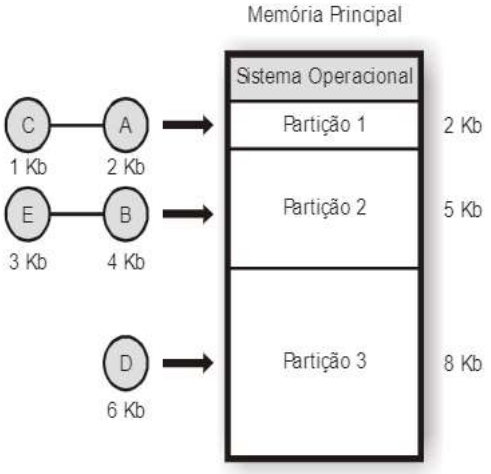
Partição	Tamanho
1	2 Kb
2	5 Kb
3	8 Kb

Programas a serem executados:





Alocação Contígua Particionada Estática Absoluta



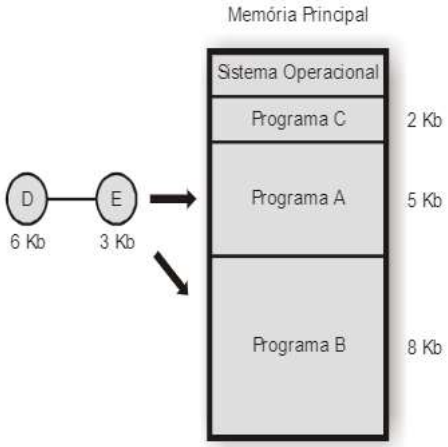
***CÓDIGO ABSOLUTO** - as referências e endereços são físicos e absolutos.*

São compilados para rodarem nas partições previamente definidas.
Só rodam na partição onde foram compilados.

Alocação da partição baseada na otimização de uso.
Alocação em função do tamanho da necessidade de memória.

A e C cabem na P1, P2 e P3
B e E não cabem na P1 e cabem na P2 e P3
D só cabe na P3

Alocação Contígua Particionada Estática Relocável



CÓDIGO RELOCÁVEL *– todas as referências e endereços são relativos ao início do programa.*

Podem ser carregados e executados em qualquer partição.

O Loader recalcula o endereçamento.

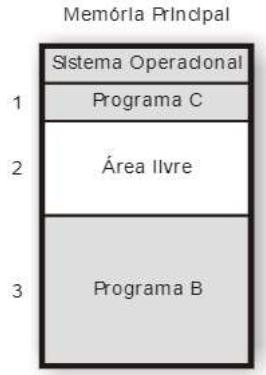
Torna a execução mais flexível.

Uso no OS/MFT Multiprogramming with Fixed Number of Tasks (IBM) do Mainframe e nos antigos UNIX.

Alocação Contígua Particionada Estática

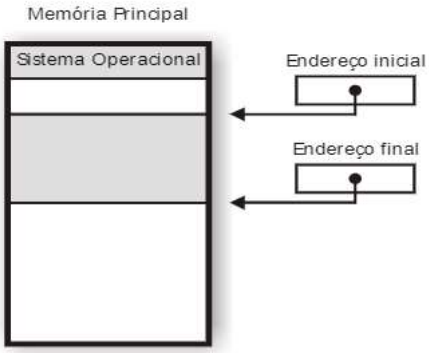
- **Tabela de Alocação de Partições**
- *O SO procura uma partição livre através da tabela de alocação para determinar a partição de execução.*
- *Detrmina o tamanho da partição e a ocupação.*

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



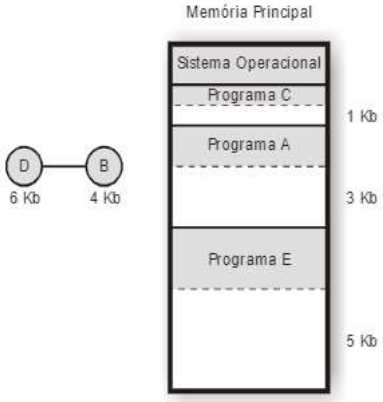
Alocação Particionada Estática

- Proteção
- **Limitação de uso – registradores de endereços**
- **Proteção contra invasão de uso indesejado**



Alocação Particionada Estática

- Fragmentação Interna
- **espaços não utilizados da partição na alocação da memória, são perdidos até o final da execução.**



- partição1 = 2 kb
- partição2 = 5 kb
- partição3 = 8 kb

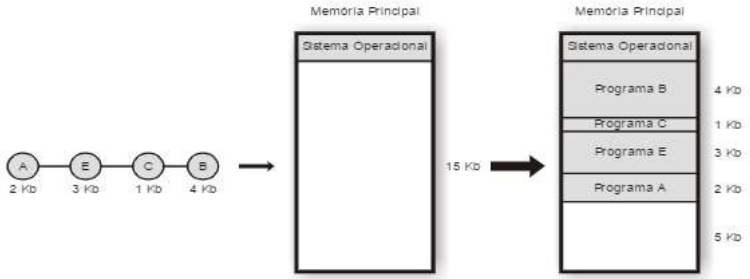
- programa C = 1 kb
- programa A = 2 kb
- programa E = 3 kb

- Fragmentação Interna
  - partição1 = 1 kb
  - partição2 = 3 kb
  - partição3 = 5 kb

Não serão usadas até o término da execução de cada partição.

Alocação Contígua Particionada Dinâmica

- na Alocação Dinâmica ou Variável, eliminação do conceito de partição de tamanho fixa.
- cada programa ocupa o espaço necessário tornando este como partição.
- não ocorre a Fragmentação Interna.



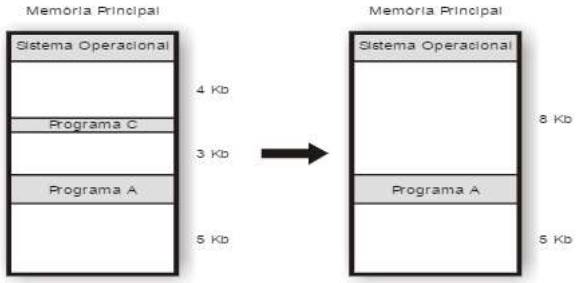
Alocação Contígua Particionada Dinâmica

- **Fragmentação Externa**
- Ocorre ao término de cada programa na Alocação Contígua Particionada Dinâmica, deixando espaços insuficientes para carga de outro programa.

O programa D tem 6 kb, espaço insuficiente nas fragmentações externas de 4 kb ou 3 kb

Alocação Contígua Particionada Dinâmica

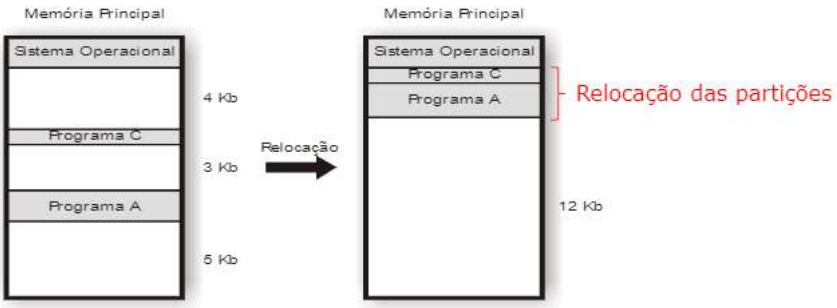
- **Solução para a Fragmentação Externa**



- Solução 1: *reunião dos espaços adjacentes liberados quando o programa **C** terminar, produzindo área livre maior de 8 Kb (4 + 1 + 3)*

Alocação Contígua Particionada Dinâmica com Relocação

- **Solução para a Fragmentação Externa**



Solução 2: *relocação de todas a partições ocupadas eliminando os espaços entre elas, produzindo uma única área livre contígua.*
- *reduz o problema de fragmentação porém aumenta complexidade*
- *Ex.: OS/MVT Multiprogramming with Variable Number of Tasks (IBM)*

Alocação Contígua Particionada Dinâmica

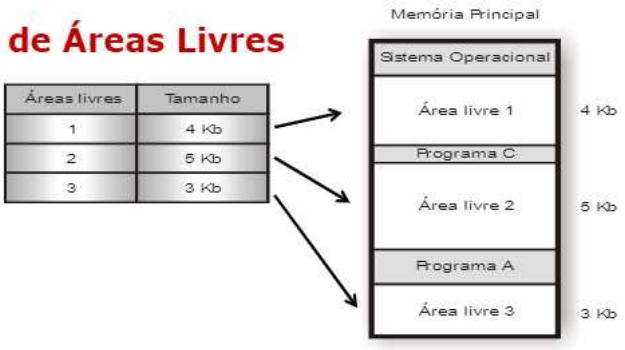
- Envolve a relocação de **todas** as partições ocupadas, eliminando **todos** os espaços entre elas e criando uma **única área livre contígua**.

  Para que esta solução possa ser implementada, é **necessário** que o *sistema tenha a capacidade de mover os diversos programas na memória principal*, ou seja, realizar a **alocação particionada dinâmica com relocação**.

Estratégias de Alocação

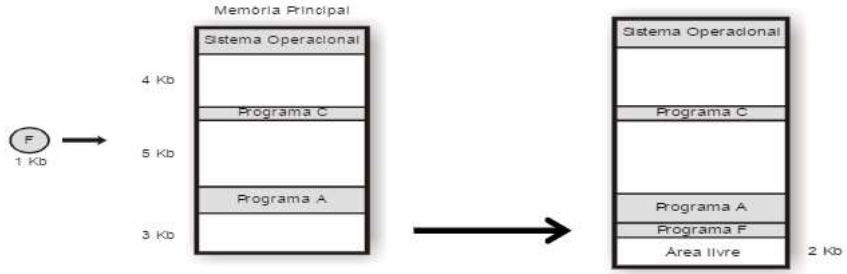
- A melhor estratégia de alocação depende de muitos fatores e condições: tamanho do programa, prioridade de execução, etc.

- **Lista de Áreas Livres**



Estratégias de Alocação

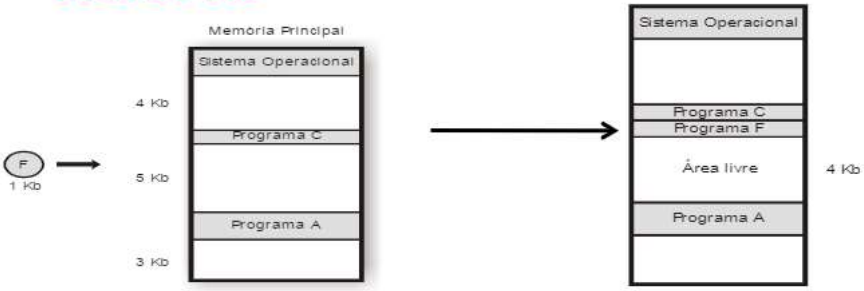
- **Best-fit**



• escolhe a partição que produza menor fragmentação interna.
• Lista de Áreas Livres ordenado pelo tamanho para diminuir tempo de busca.
• desvantagem: tendência de aumentar a quantidade de fragmentação externa de pequeno tamanho, prejudicando as novas alocações.

Estratégias de Alocação

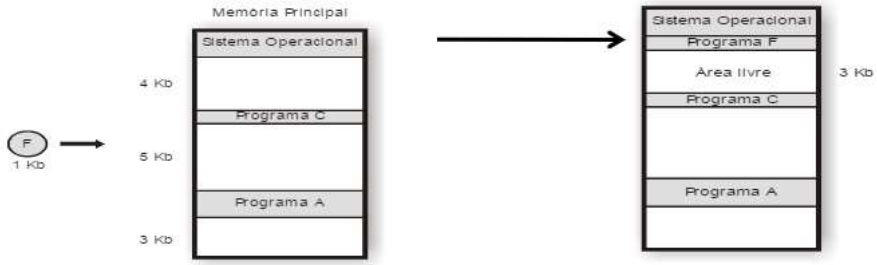
- **Worst-fit**



• escolhe a partição que produza maior Fragmentação Interna
• tendência de utilização de espaços de memória de maior tamanho, sendo possível a reutilização, diminuindo os problemas de Fragmentação Externa.
• desvantagem: por utilizar os maiores espaços de memória disponíveis, a fragmentação interna é grande, qdo. executa programas de pequeno porte, ficando poucas chances para programas maiores.

Estratégias de Alocação

• First-fit

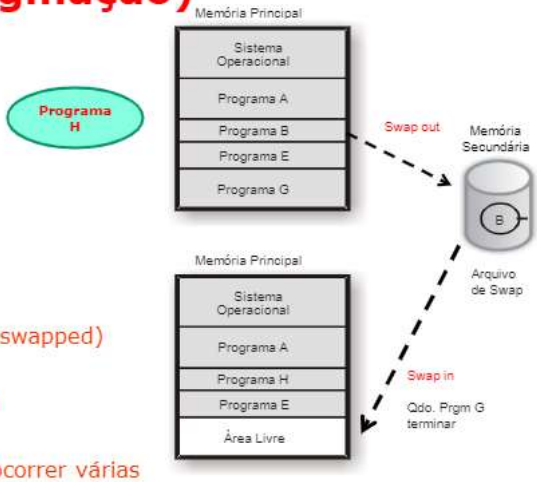


• escolhe a primeira partição livre de tamanho suficiente para carregar o programa.
• Lista de Áreas Livres está ordenada por endereços crescentes.
• utiliza as áreas livres de endereços mais baixos, existe grande chance de obter grande partição livre nos endereços mais alta.
• É a estratégia mais rápida por consumir menos recursos do sistema.

SWAPPING

• Swapping (Paginação)

• Resolver problemas de insuficiência de memória principal
• Atender processos no "Estado de Espera" da memória livre
• **Algorítmo de escolha:** menor chance de execução no "Estado de Espera (Espera Outswapped) ou Pronto (Pronto Outswapped)"
• Loader com Relocação Dinâmica Implementado.
• o Processo de SWAPPING pode ocorrer várias Vezes para o mesmo programa



SWAPPING

- Relocação Dinâmica
- Registrador de Relocação recebe o endereço inicial da carga.
- Registrador de Relocação + Endereço da instrução = ender. Físico.
- O programa pode ser carregado em qualquer posição da memória.
- Permite maior compartilhamento da Memória Principal e maior utilização dos recursos computacionais.
- desvantagem: elevado custo do dispositivo de E/S (swap in/out).
- Com pouca memória disponível e em situações críticas, o sistema pode ficar quase que dedicado a Swapping deixando de executar processos residentes.
- 1960 – IBM OS/360 da IBM e CTSS da MIT.