Sistemas Operacionais

Professores: Cristiano Bonato Both

Sumário

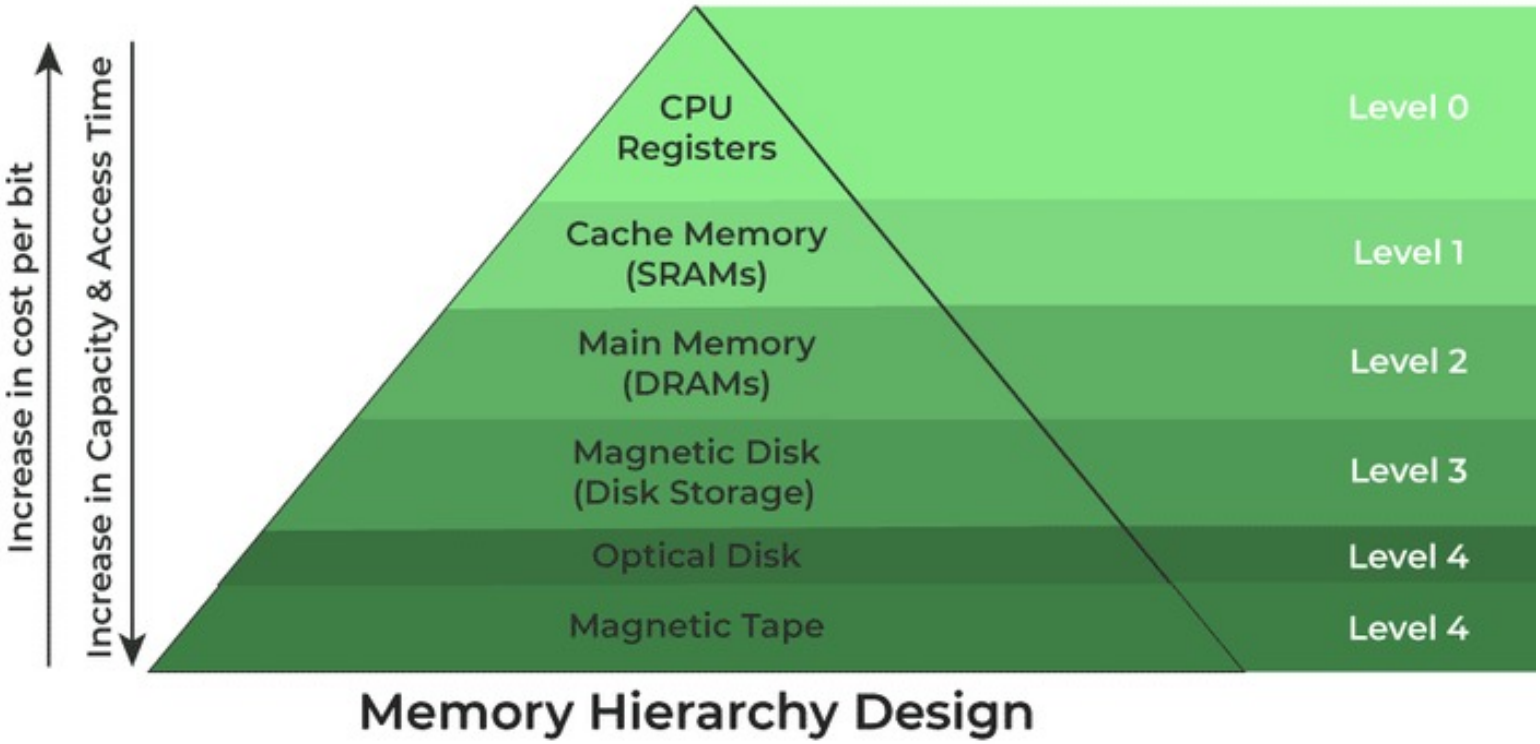
- Gerência de memória
  - Memória Lógica
  - Memória Física
  - Mecanismos de alocação
  - Algoritmos de alocação
  - *Swapping*
- Referências

Introdução

- Multiprogramação implica em manter vários processos em memória
- Memória necessita ser alocada de forma eficiente
- Existem diferentes técnicas para gerência de memória
  - Dependem do *hardware* do processador

Considerações Gerais

- Um sistema de memória deve ser visto de forma hierárquica



Memória Lógica e Física

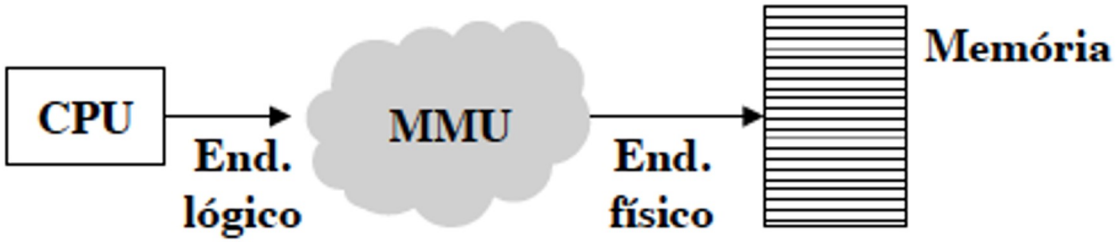
- Memória lógica
  - É aquela que o processo “enxerga”, *i.e.*, pode acessar
  - Endereços lógicos são manipulados por um processo
- Memória física
  - Implementada pelos circuitos integrados de memória
  - Endereços físicos correspondem a uma posição real de memória

Endereço Lógico *versus* Físico

- Espaço lógico de um processo é diferente do espaço físico
  - Endereço lógico: gerado pela CPU (endereço virtual)
  - Endereço físico: endereços enviados para a memória RAM
- Endereços lógicos são transformados em endereços físicos no momento de execução dos processos

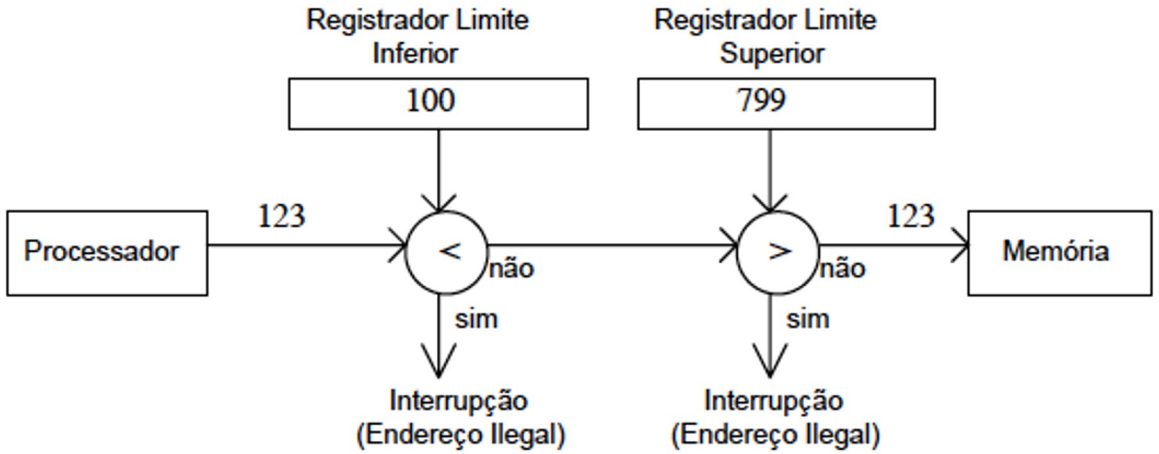
Unidade de Gerência de Memória

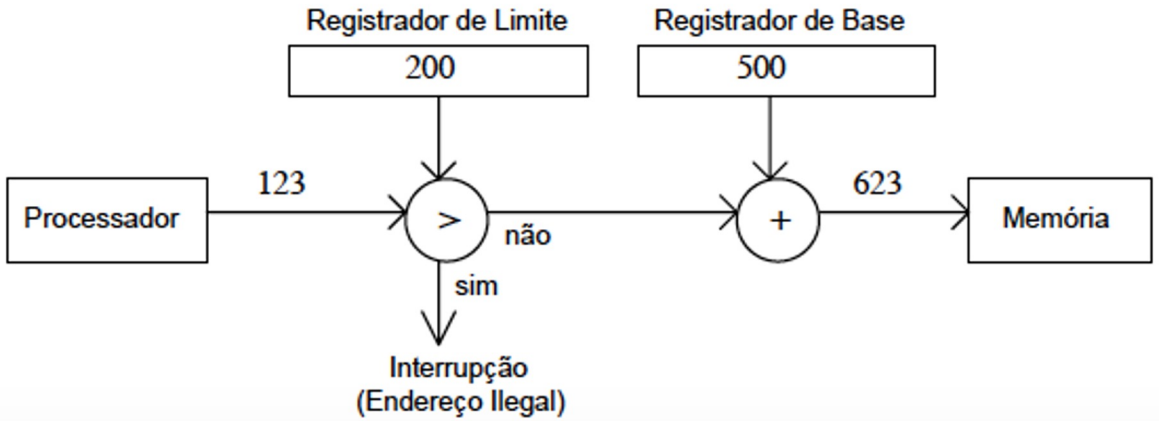
- *Memory Management Unit* (MMU)
- *Hardware* que faz o mapeamento entre endereço lógico e endereço físico



- Complexidade variável:
  - Mecanismos de suporte para proteção, carga de programas, tradução de endereços lógicos para endereços físicos, etc.

Exemplos de MMU





Execução de Programas

- Um programa deve ser transformado em um processo para poder ser executado. É necessária a alocação:
  - de um descritor de processos
  - de áreas de memória para código, dados e pilha

- Transformação é feita através de uma série de passos
  - Compilação, diretivas de compilação e/ou montagem, ligação, etc.

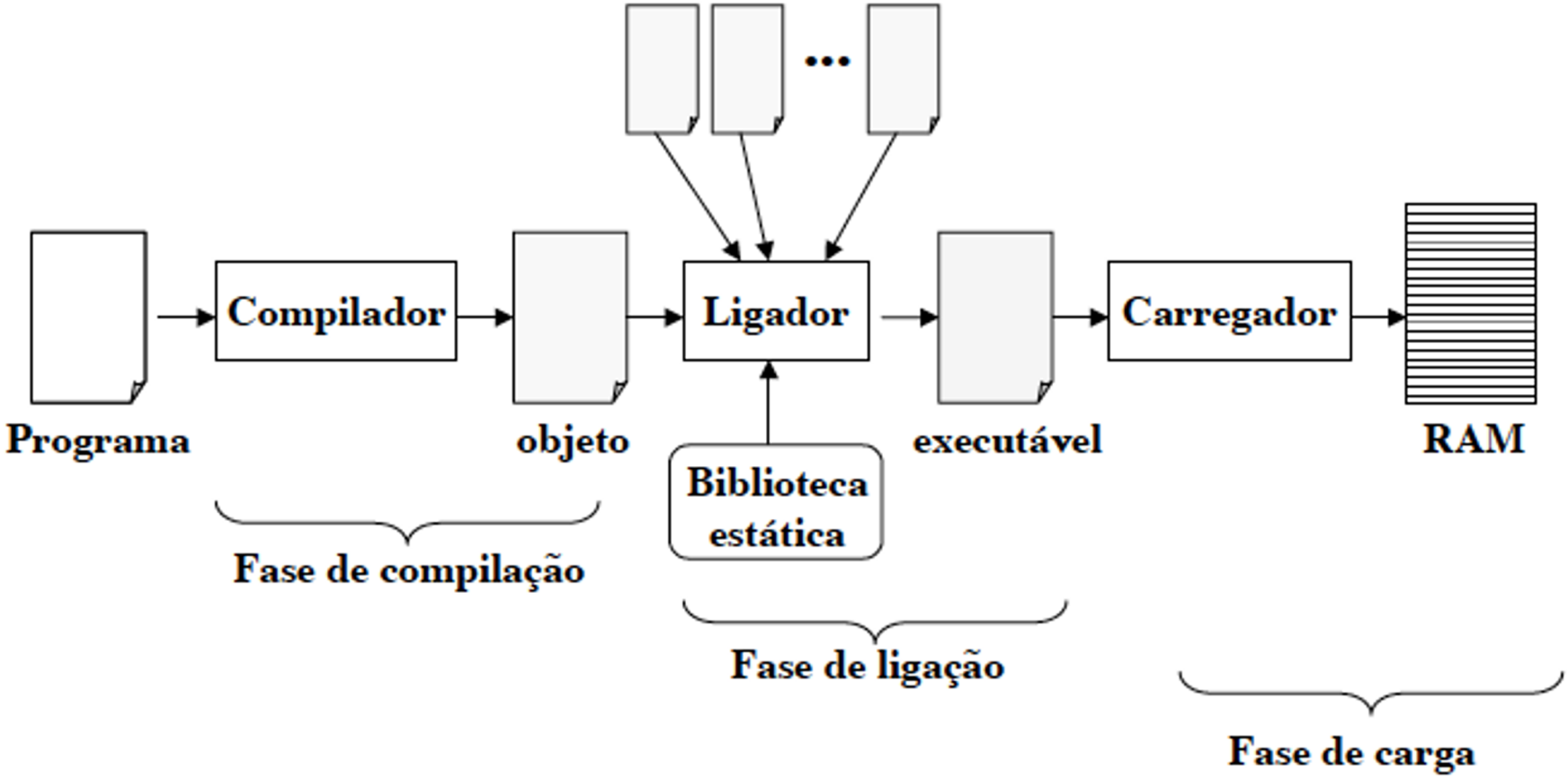
- Amarração de endereços (*binding*)

Amarração de Endereços (*binding*)

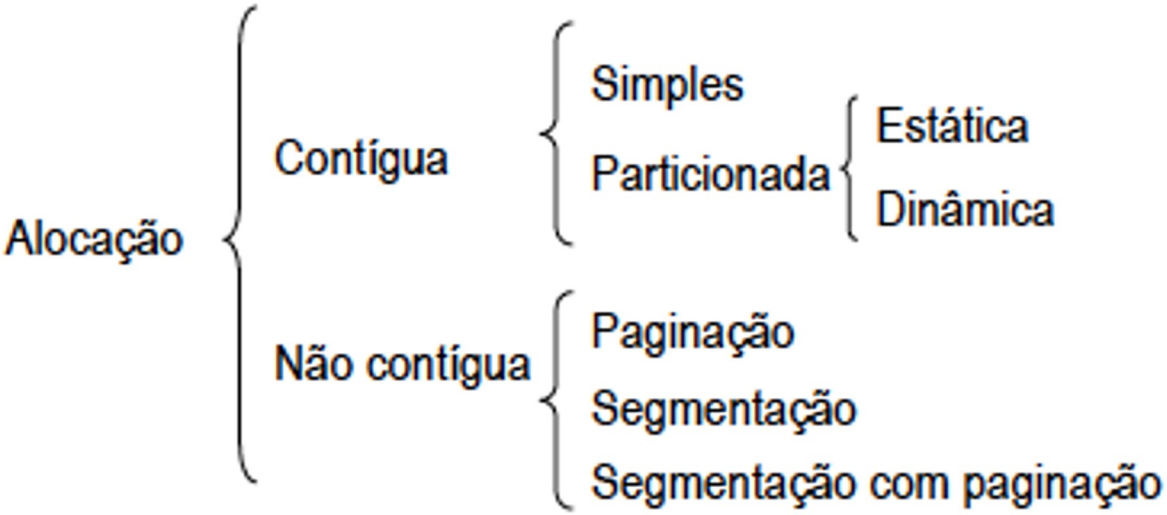
- Atribuição de endereços (posições de memória) para código e dados pode ser feita em três momentos:
  - Em tempo de compilação
  - Em tempo de carga
  - Em tempo de execução

- Diferenciação entre o endereço lógico e o endereço físico
  - Como traduzir endereço lógico em endereço físico
  - Código absoluto e código relocável

Transformação de Programa em Processo



Mecanismo para Alocação de Memória



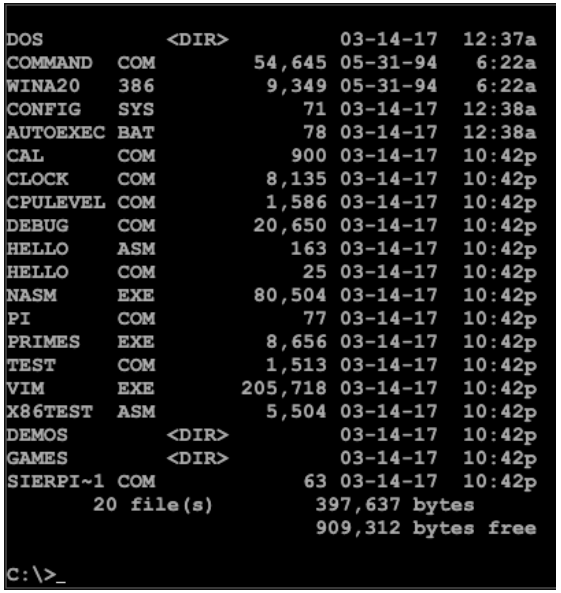
- Até memória virtual, supor que para um programa ser executado, o programa necessita estar carregado completamente em memória

Alocação Contígua Simples

- Memória principal é dividida em duas partições:
  - Sistema operacional (parte baixa da memória)
  - Processo do usuário (restante da memória)

- Usuário tem controle total da memória, podendo inclusive acessar a área do sistema operacional
  - *e.g.*, DOS (não confiável)
  - Evolução:
    - Inserir proteção através de mecanismos de *hardware* + *software*
      - Registradores de base e limite
      - *Memory Management Unit* (MMU)

https://copy.sh/v86/?profile=msdos

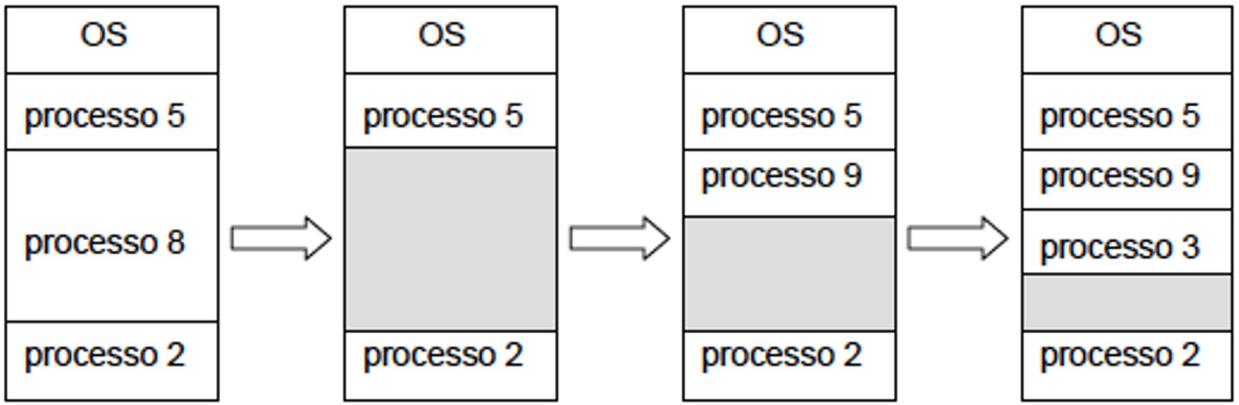


Alocação Contígua Particionada

- Existência de múltiplas partições
- Imposta pela multiprogramação
- Filosofia:
  - Dividir a memória em blocos (partições)
  - Cada partição pode receber um processo (programa)
  - Grau de multiprogramação é fornecido pelo número de partições
    - Não considerando a existência de *swapping*
- Duas formas básicas:
  - Alocação contígua com partições fixa (estática)
  - Alocação contígua com partições variáveis (dinâmica)

Alocação Contígua Particionada

- O sistema operacional é responsável pelo controle das partições mantendo informações como:
  - Partições alocadas
  - Partições livres
  - Tamanho das partições



Alocação Contígua Particionada Fixa

- Memória disponível é dividida em partições de tamanho fixo que podem ser do mesmo tamanho ou não
  - Processos podem ser carregados em qualquer partição?
  - Número de processos que podem estar em execução ao mesmo tempo
    - Sem *swapping*: igual ao número de partições (máximo)
    - Com *swapping*: maior que número de partições
  - Programa é maior que o tamanho da partição
    - Não executa a menos que se empregue um esquema de *overlay*

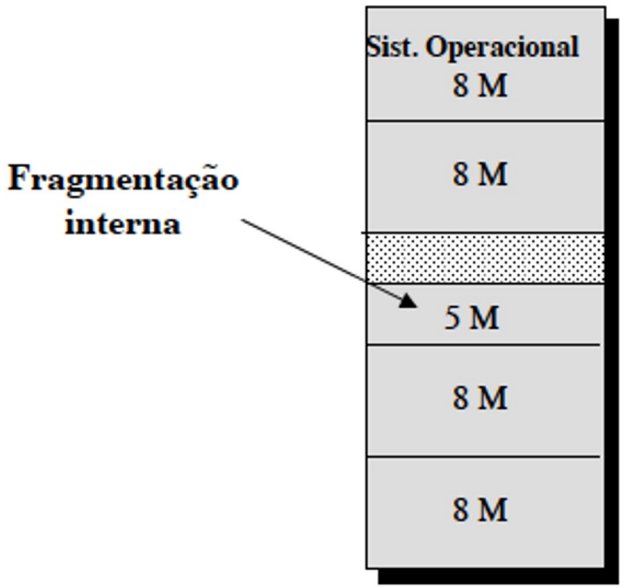
Gerenciamento de Partições Fixas

- Com código absoluto
  - Um processo só pode ser carregado na área de memória (partição) para a qual foi compilado
  - Pode haver disputa por uma partição mesmo tendo outros livres
    - Processo é mantido no escalonador de longo prazo
    - Empregar swapping

- Com código relocável
  - Um processo de tamanho menor ou igual ao tamanho da partição pode ser carregado em qualquer partição disponível
  - Se todas as partições estão ocupadas, duas soluções:
    - Processo é mantido no escalonador de longo prazo
    - Empregar swapping (escalonamento a médio prazo)

Fragmentação Interna

- Problema da alocação fixa é uso ineficiente da memória principal
- Um processo, não importando quão pequeno seja, ocupa uma partição interna
  - Fragmentação interna

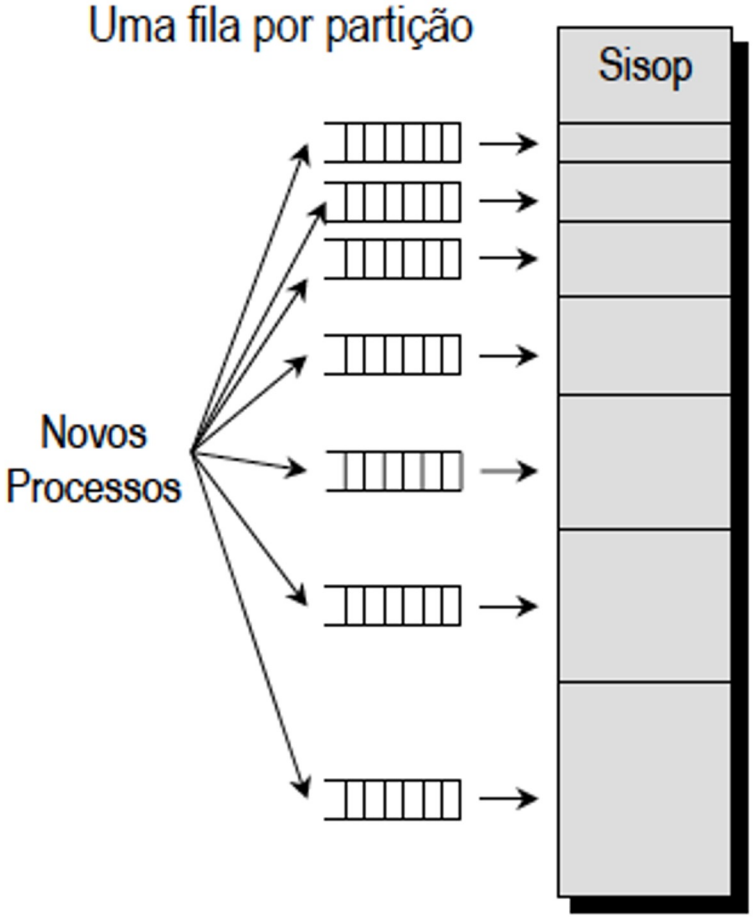


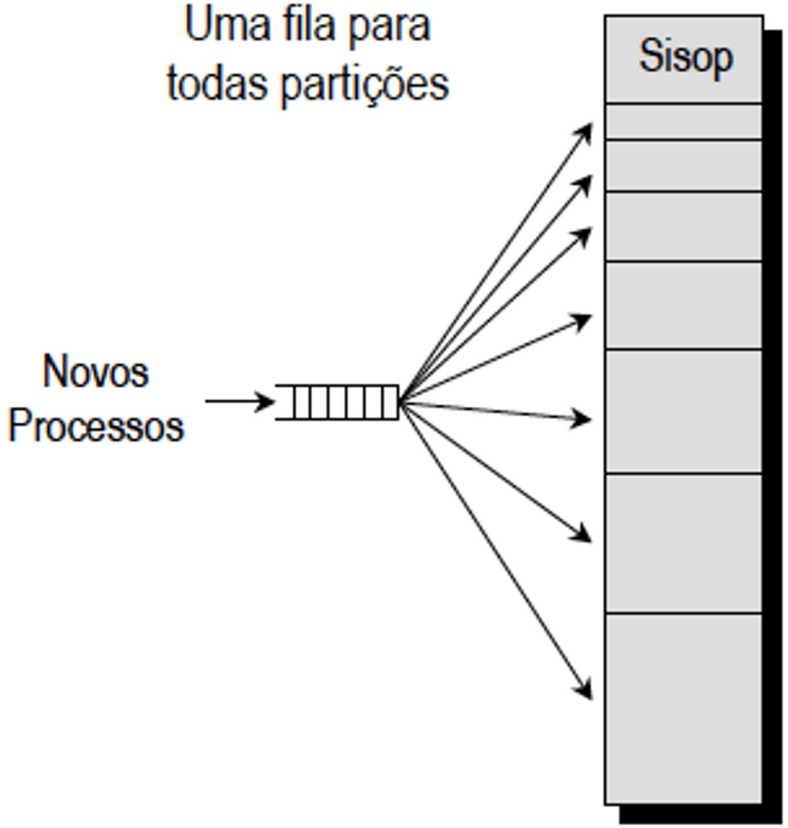
Algoritmos para Alocação de Partições Fixas

- Se código é absoluto a alocação é determinada na fase de montagem, compilação ou ligação

- Se código é relocável:
  - Partições de igual tamanho
  - Partições de diferentes tamanhos
    - Atribui o processo à menor partição livre capaz de armazená-lo, para minimizar o desperdício de memória

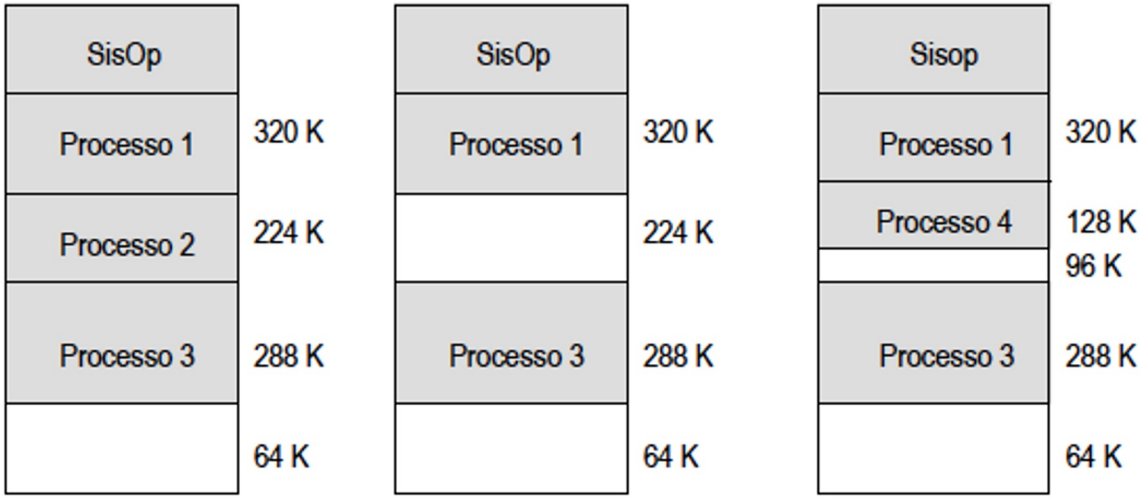
Algoritmos para Alocação de Partições Fixas





Alocação Particionada Dinâmica

- Objetivo é eliminar a fragmentação interna
- Processos alocam memória de acordo com suas necessidades
- Partições são em número e tamanho variáveis

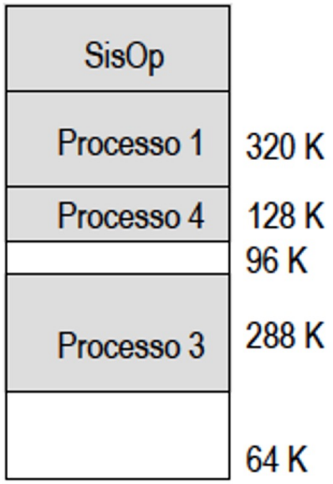


Fragmentação Externa

- A execução de processo pode criar pedaços livres de memória
  - Pode haver memória disponível, mas não contígua
    - Fragmentação externa

Exemplo:

Criação processo 120K



Soluções Possíveis – Fragmentação Externa

- Reunir espaços adjacentes de memória
- Empregar compactação
  - Desvantagem:
    - Consumo do processador
    - Acesso à disco
- Acionado somente quando ocorre fragmentação
- Necessidade de código relocável

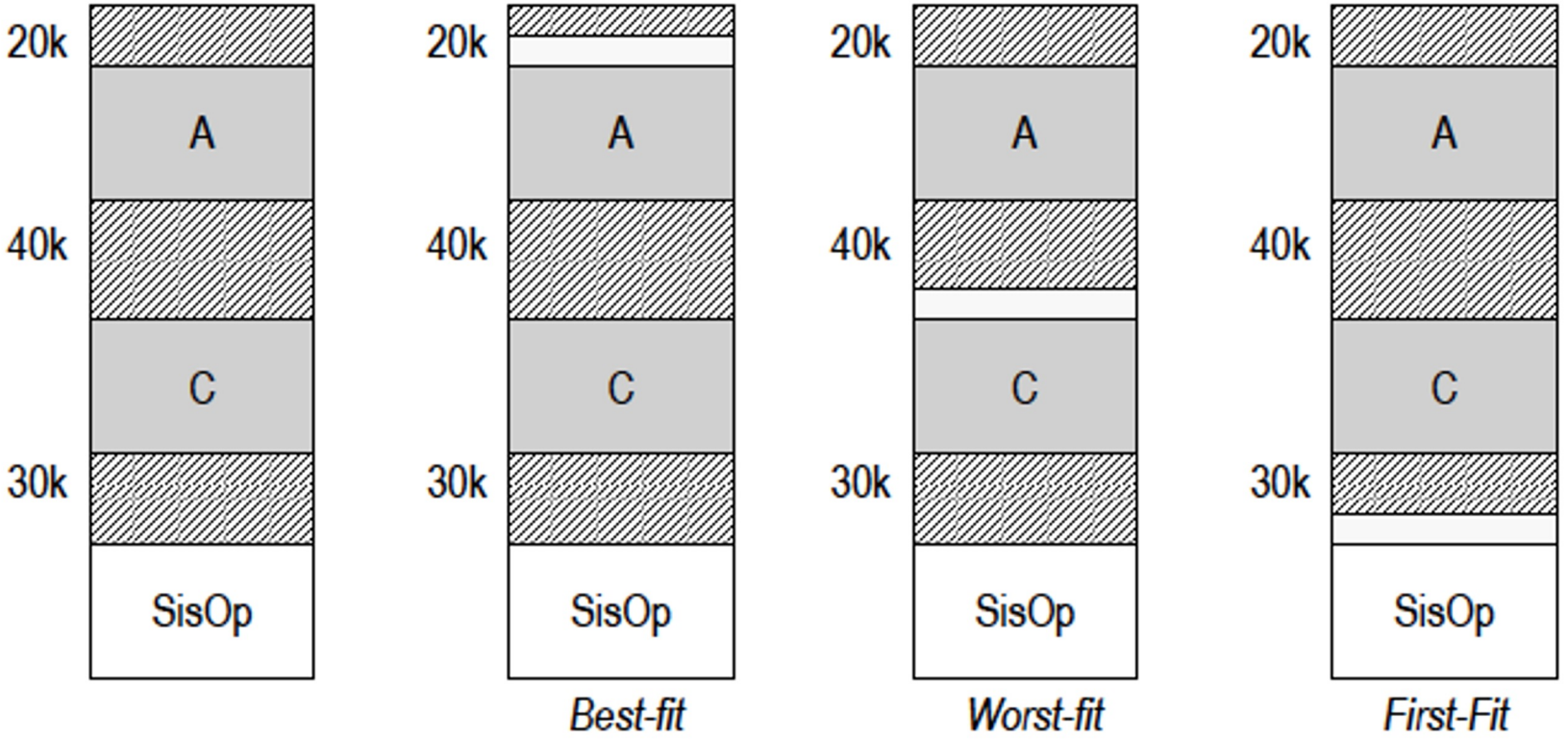
Gerenciamento de Partições Dinâmicas

- Determinar qual área de memória livre será alocada a um processo
- Sistema operacional mantém uma lista de lacunas
  - Pedaços de espaços livres em memória
- Necessidade de percorrer a lista de lacunas sempre que um processo é criado
  - Como percorrer essa lista?

Algoritmos para Alocação Contígua Dinâmica

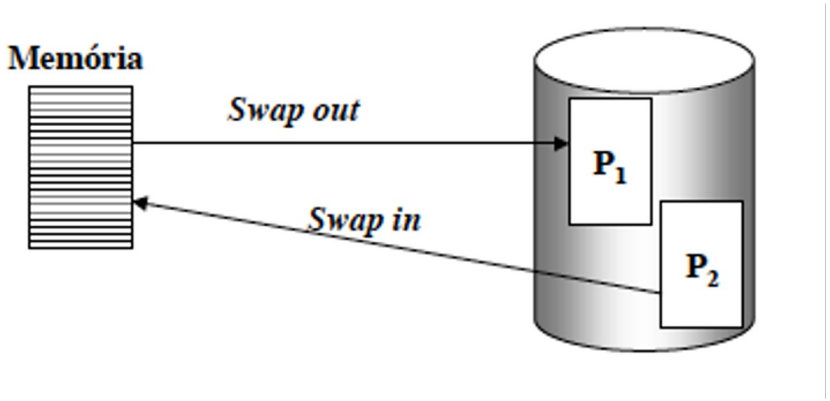
- *Best fit*
  - Minimizar tam_processo – tam_bloco
  - Deixar espaços livres os menores possíveis
- *Worst fit*
  - Maximizar tam_processo – tam_bloco
  - Deixar espaços livres os maiores possíveis
- *First fit*
  - tam_bloco > tam_processo
- *Circular fit*
  - Variação do *first fit*

Exemplos


20k
A
40k
C
30k
SisOp
20k
A
40k
C
30k
SisOp
Best-fit
20k
A
40k
C
30k
SisOp
Worst-fit
20k
A
40k
C
30k
SisOp
First-Fit

Swapping

- Processo precisa estar na memória para ser executado
  - Se não há mais espaço em memória é necessário fazer um rodízio de processos em memória



- Memória secundária suficientemente grande para armazenar cópias de todos os processos de usuários

Swapping

- Tempo de *swap* é proporcional ao tamanho do processo
  - Possui influência na troca de contexto
- Processos que realizam E/S
  - Nunca realizar *swap* em processos que estão com E/S pendente
  - Utilizar *buffers* de E/S internos ao sistema
- Existem variantes do sistema de *swapping* utilizados em sistemas como UNIX ou Windows

Referências Bibliográficas

- SILBERSCHATZ, A.; GALVIN, Peter; GAGNE Greg, Operating System Concepts Essentials. John Wiley & Sons, Inc. 2th edition, 2013.
- TANENBAUM, Andrew S. Sistemas operacionais modernos. 3a. ed. São Paulo: Pearson, 2009-2013. p. 653.
- OLIVEIRA, Rômulo; CARÍSSIMI, Alexandre; TOSCANI, Simão. Sistemas Operacionais. Porto Alegre: Bookman, 4a. ed. 2010.