Gerenciamento de Memória Principal Sistemas Operacionais

Charles Tim Batista Garrocho

Instituto Federal do Paraná – IFPR Campus Goioerê

charles.garrocho.com/SO2016

 $\verb|charles.garrocho@ifpr.edu.br| \\$

Técnico em Informática



Conceitos Básicos

A CPU pode ser compartilhada por um conjunto de processos, **otimizando** o *tempo de resposta* do computador para seus usuários.

Otimização da CPU requer vários processos em **memória**, isto é, a memória deve ser **compartilhada**.

O gerenciamento de memória varia entre diversas **estratégias** de alocação de memória.

A escolha de uma estratégia depende do **projeto** de *hardware* do sistema.



Alocação de Memória

Um programa reside em um disco como um arquivo binário executável.

Para que seja executado, o programa deve ser *carregado* na **memória principal** e inserido em um processo.

A coleção de processos em disco, em espera para serem carregados em memória para execução, forma a **fila de entrada**.

O procedimento normal é **selecionar** um dos processos na fila de entrada e *carregá-lo* na memória. Ao ser **finalizado** o seu espaço de memória é declarado **disponível**.



Vinculação de Endereços

A maioria dos sistemas permite que um processo de usuário *resida* em qualquer parte da **memória física**.

Embora os espaços de *endereçamento* do computador inicie em 00000, o primeiro endereço do processos do usuário não precisa ser 00000.

Os endereços no programa-fonte (Python) são geralmente simbólicos.

O compilador vincula os endereços simbólicos a endereços relocáveis.

O **carregador** por sua vez vincula os endereços relocáveis a endereços *absolutos*.



Espaço de Endereçamento Lógico Versus Físico

Um endereço *gerado* pela CPU é normalmente chamado de **endereço lógico**, ele é gerado durante a *compilação*.

Através da *realocação dinâmica* que consiste em utilizar um *endereço base* (**endereço físico**) e os endereços lógicos como *offset*, obtem-se o endereço físico para cada endereço lógico.

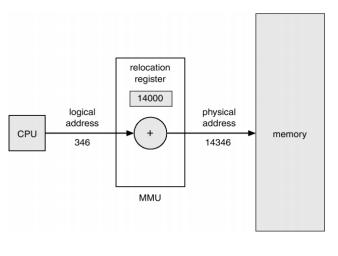
Sendo o endereço físico um endereço que representa uma **localização** real e valida na memória.

O mapeamento de endereços lógicos em endereços físicos, em tempo de execução, é realizado por um dispositivo de hardware chamado **unidade** de gerenciamento de memória (MMU - memory management unit).



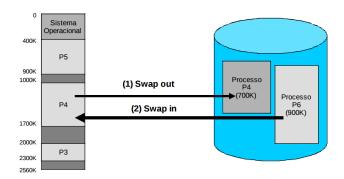
Relocação Dinâmica Utilizando um Registrador

Endereços lógicos são transformados.



Swapping

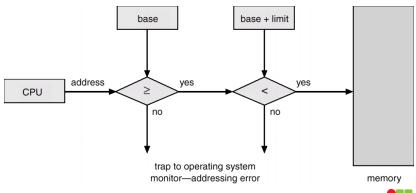
Processos suspensos **fora** da memória, exigem relocação ao serem **recarregados**.



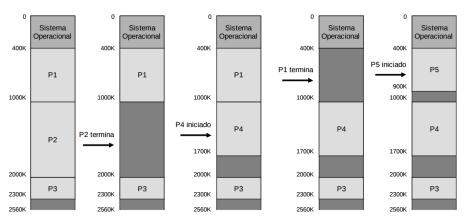


Proteção de Memória

Todo acesso é comparado à faixa definida pelos registradores **base** e **limite**. Acessos fora da faixa geram *TRAP*.



Alocação com Múltiplos Processos





Problema de Alocação Dinâmica

Onde colocar um processo novo?

O ideal é utilizar um dos algoritmos de alocação de memória abaixo:

- First-fit (primeiro apto): primeiro que couber;
- Best-fit (mais apto): o de tamanho mais próximo;
- Worst-fit (menos apto): sempre o maior buraco;

First-fit e best-fit se saem melhor que worst-fit em termos de velocidade e utilização do espaço.



Fragmentação e Compactação

Fragmentação é o desperdício de espaço disponível em memória.

O esquema de **particionamento** pode ser *dinâmico* ou *fixo* na memória principal.

A fragmentação interna é a perda de espaço dentro de uma área de tamanho fixo.

A fragmentação externa acontece quando os programas terminam e deixam espaços cada vez menores na memória, não permitindo o ingresso de novos programas.

A Solução para a fragmentação externa é a **Compactação**, onde é movido blocos ocupados para perto uns dos outros, e agrupados em um *único* bloco maior.

Exercícios

- Explique a diferença entre endereços lógicos e físicos.
- Explique a diferença entre fragmentação interna e externa.
- O que é swapping e como funciona?
- Dadas partições de memória com 100K, 500K, 200K, 300K e 600K (em ordem), de que forma cada um dos algoritmos do primeiro-apto, mais-apto e menos-apto alocarão processos com 212K, 417K, 112K e 426K (em ordem)? Qual dos algoritmos faz uso mais eficiente da memória?

