Sistemas Operacionais: Gerenciamento de Memória

Lucas Rickelme Araújo Costa
Sabrina Carlos Costa

1º Explique a diferença entre alocação contígua e não contígua.

Na alocação contígua, cada processo obtém uma sequência única de endereços da memória que estão juntos e continuos. Isso facilita a implementação de algoritmos de substituição de páginas e torna a execução dos processos mais eficiente, já que não há necessidade de deslocamento entre diferentes partes do processo ao executar instruções ou acessar dados. No entanto, essa abordagem também pode levar à fragmentação externa, onde o espaço disponível na memória é dividido em várias pequenas lacunas que são insuficientes para acomodar novos processos, mesmo que a soma total dessas lacunas seja maior do que o tamanho necessário.
Por outro lado, na alocação não contígua, os blocos de memória são atribuídos aos processos sem necessariamente estarem fisicamente próximos. Isso é mais comumente implementado em sistemas que usam páginas ou segmentos para dividir a memória e permitir maior flexibilidade no gerenciamento. A alocação não contígua pode evitar a fragmentação externa, mas isso introduz novos desafios, como o overhead de gerenciar múltiplas partes do processo e o potencial aumento da latência durante os acessos à memória.

2º Descreva os tipos de fragmentação e como eles afetam o desempenho do sistema.

Existem dois principais tipos de fragmentação: a fragmentação interna e a fragmentação externa. A fragmentação interna ocorre quando um bloco de memória é maior do que o processo precisa, mas não há espaço disponível em outro local da memória para acomodar um bloco menor. Isso significa que parte do bloco alocado é desperdiçada e não pode ser reutilizada por outros processos. A fragmentação interna pode levar a uma sobrecarga na memória, pois o espaço desalocado ainda está marcado como ocupado, impedindo o sistema de usar efetivamente toda a capacidade disponível.
A fragmentação externa acontece quando há muitos pequenos espaços livres dispersos pela memória que não são grandes o suficiente para acomodar novos processos, mesmo que a soma total desses espaços disponíveis seja maior do que o tamanho necessário. Isso pode dificultar a alocação de novos processos e levar à falta de memória disponível em um

sistema que na verdade possui recursos suficientes. A fragmentação externa também pode aumentar a latência, pois o sistema precisa procurar por espaços contíguos adequados para alocar novos processos.
Ambas as formas de fragmentação podem afetar significativamente o desempenho do sistema ao impedir uma utilização eficiente da memória. Isso pode levar a sobrecarga na CPU, aumento da latência, diminuição da capacidade de multitarefas e, em casos extremos, até falhas no sistema.

3º Compare e contraste paginação e segmentação.

A paginação divide o processo em partes iguais chamadas de páginas, que são então alocadas individualmente na memória física ou virtual. A principal vantagem da paginação é que ela permite que processos grandes sejam facilmente gerenciados e também facilita a implementação de mecanismos de substituição de página para lidar com situações de falta de memória. No entanto, essa abordagem pode levar à fragmentação externa se os espaços livres na memória forem muito pequenos para acomodar novas páginas.
Por outro lado, a segmentação divide o processo em partes chamadas de segmentos com base em funções semelhantes dentro do programa. Isso pode ser mais eficiente em termos de uso da memória, já que cada segmento pode ser alocado conforme necessário e não precisa ser preenchido completamente. A segmentação também facilita o compartilhamento de dados entre processos, pois diferentes partes do mesmo processo podem ser alocadas na mesma parte da memória. No entanto, essa abordagem pode levar à fragmentação interna se os segmentos tiverem tamanhos que não são múltiplos do tamanho da página, e também pode exigir mais esforço de gerenciamento em comparação com a paginação.

4º Qual é a função da tabela de páginas em um sistema de paginação?

A tabela de páginas é uma estrutura crucial em sistemas que usam paginação para gerenciar memória. Ela é associada a cada processo e contém informações sobre as páginas do processo e onde essas páginas estão localizadas na memória física ou virtual.
Cada entrada na tabela de páginas corresponde a uma página específica do processo e geralmente inclui informações como o número da página (endereço lógico), o quadro (endereço físico) onde essa página está armazenada, indicadores de estado como se a página foi modificada desde sua última leitura ou escrita e bits de controle para gerenciar eventos como referências à memória.
A tabela de páginas é usada pelo sistema operacional para mapear endereços lógicos de um processo para seus respectivos endereços físicos na memória. Durante a execução do programa, o processador usa essa tabela para traduzir os endereços lógicos em acessos à memória física ou virtual. Isso é essencial para permitir que diferentes processos compartilhem a mesma área de memória e para suportar mecanismos como paginação demanda-lazy, onde as páginas são carregadas na memória apenas quando realmente necessárias.

5° Como a segmentação pode melhorar a organização da memória em comparação com a paginação?

A segmentação pode oferecer uma melhor organização da memória do que a paginação de várias maneiras:

Tamanhos dinâmicos: A segmentação permite que os segmentos tenham tamanhos variáveis, o que é mais flexível do que a paginação, onde todas as páginas têm tamanho fixo.

Compartilhamento de código: Diferentes processos podem compartilhar facilmente o mesmo segmento de código se eles executarem a mesma função ou programa. Isso economiza memória e reduz a sobrecarga associada ao carregamento repetitivo do mesmo código em diferentes áreas da memória.

Melhor gerenciamento de espaço: A segmentação pode ajudar a evitar a fragmentação externa mais eficazmente, pois os processos podem ser organizados na memória com base no tamanho dos seus segmentos, reduzindo o número de lacunas menores.

Facilidade de alocação e desaloção: A segmentação torna a alocação e desalocação de memória mais fáceis, especialmente para processos grandes que podem ser facilmente fragmentados em vários segmentos.

Escalonamento e proteção: A segmentação também permite um controle melhor sobre o acesso à memória, facilitando a implementação de mecanismos como escalação (a capacidade de permitir ou restringir processos a acessar certas áreas da memória) e proteção de memória.

Em suma, embora tanto a paginação quanto a segmentação tenham suas próprias vantagens, a segmentação pode oferecer uma organização mais eficiente da memória, especialmente para sistemas que requerem flexibilidade no tamanho dos dados e códigos, ou precisam compartilhar código entre vários processos.

Parte Prática

Repositório no github:

https://github.com/csabrina/so-pratica-memoria

Implementação:

Para a implementação do simulador de gerenciamento de memória usando paginação, utilizamos Java 17 e Maven. Adotamos o princípio da responsabilidade única e a modularização. O código foi dividido em classes que representam a memória física, a tabela de páginas, o processo e o algoritmo de substituição FIFO.

Estrutura do Projeto:

- MemoryFrame.java: Representa um quadro de memória física.
- PageTable.java: Representa a tabela de páginas de um processo.
- Process.java: Representa um processo com seu espaço de endereçamento lógico.
- FIFOReplacement.java: Implementa o algoritmo de substituição FIFO.

- MemoryManager.java: Gerencia a memória física e a alocação de páginas.
- Main.java: Classe principal para executar o simulador.

Passo a passo para execução:

- Certifique-se de ter o JDK 17 instalado.
- Compile todos os arquivos Java usando o comando:
    - javac *.java ou mvn install
- Execute o programa principal com o comando:
    - java Main

Detalhes do comportamento de execução do simulador:

- O programa inicializa a memória física com 16 quadros.
- Cria dois processos, cada um com 4 páginas.
- Aloca as páginas dos processos na memória física, substituindo páginas usando o algoritmo FIFO quando necessário.
- Exibe o estado da memória física e as tabelas de páginas após cada alocação.

Cenário de Execução:

- **Alocação do Processo 1:**
    - O Processo 1 tem 4 páginas (Página 0, Página 1, Página 2, Página 3).
    - As páginas são alocadas nos primeiros 4 quadros livres (Quadros 0, 1, 2, 3).
    - O estado da memória física e da tabela de páginas é exibido.
- **Alocação do Processo 2:**
    - O Processo 2 também tem 4 páginas (Página 0, Página 1, Página 2, Página 3).
    - As páginas são alocadas nos próximos 4 quadros livres (Quadros 4, 5, 6, 7).
    - O estado da memória física e da tabela de páginas é exibido.