Leonardo Vecchi Meirelles - J201ECP002

Questão 1) - Capítulo 14

• Memory leaks: isso ocorre quando a memória que foi alocada dinâmicamente usando malloc() não é liberada usando free() quando não é mais necessária. Isso pode fazer com que o programa fique sem memória, resultando em falhas ou comportamentos inesperados.

• Double Free Errors: ocorre quando um bloco de memória que já foi liberado usando free() é posteriormente liberado novamente. Isso pode causar corrupção de memória e comportamentos indefinidos.

• Buffer Overflows e Underruns: ocorrem quando um programa grava dados fora dos limites de um buffer de memória, podendo causar corrupção de memória.

• Memory Fragmentation: ocorre quando muitos pequenos blocos de memória são alocados e liberados de forma a deixar o layout da memória fragmentado, com muitos pequenos intervalos de memória não utilizada entre os blocos. Isso pode levar ao uso ineficiente de memória e dificultar a alocação de grandes blocos de memória.

Leonardo Vecchi Meirelles - 1201ECP002

Cap. 15 - Questão 1)

Realocação dinâmica é um método para implementar a conversão de endereço no qual o endereço de memória física de um processo é determinado em tempo de execução pelo SO.

Ela funciona atribuindo a cada processo um identificador exclusivo, conhecido como endereço base, que é usado para calcular o endereço de memória física de cada referência de memória feita pelo processo. Quando um processo faz uma referência à memória, o SO adiciona o endereço base ao endereço virtual especificado pelo processo para determinar o endereço da memória física correspondente.

Para suportar a realocação dinâmica, o hardware deve fornecer um mecanismo para interceptar as referências à memória feitas pelo processo e encaminhá-las ao sistema operacional. Esse mecanismo geralmente é fornecido por meio do uso de uma unidade de gerenciamento de memória (MMU), que fica entre o processador e a memória e é responsável por traduzir endereços virtuais em endereços físicos.

Leonardo Vecchi Meirelles - 12011ECP002
Cap. 15 - Questão 2)

1- O sistema operacional cria um novo processo e atribui a ele um identificador exclusivo, o endereço base.

2- O sistema operacional carrega o código e os dados do processo na memória física, começando em um local determinado pelo endereço base.

3- O processo inicia a execução e, ao fazer uma referência à memória, a referência é interceptada pelo hardware e encaminhada ao sistema operacional para tradução.

4- O sistema operacional usa o endereço base para calcular o endereço da memória física correspondente ao endereço virtual especificado pelo processo.

5- O endereço da memória física é retornado ao hardware, que completa a referência à memória e permite que o processo continue em execução.

Leonardo Vecchi Meirelles - 12011ECP002

Cap. 16 - Questão 1)

A razão para usar a segmentação é fornecer uma maneira mais flexível e eficiente de gerenciar a memória do que o modelo tradicional de memória plana usado pela maioria dos primeiros sistema.

Com um modelo de memória segmentada, um programa pode alocar memória em blocos de tamanhos variados, em vez de ficar limitado a páginas ou frames de tamanho fixo. Isso permite que os programas usem a memória com mais eficiência e utilizem melhor o espaço de endereçamento disponível. Além disso, o uso de segmentos pode ajudar a fornecer melhor proteção e isolamento de memória entre diferentes partes do programa, facilitando a prevenção de interferências acidentais ou maliciosas entre código e dados

Leonardo Vecchi Meirelles - 12011ECP002
Cap. 16 - Questão 2)

Na segmentação, um endereço lógico é composto por um número de segmento (segment number) e um deslocamento dentro desse segmento (offset). O número do segmento é usado para consultar a tabela de descritor de segmento (SDT) para obter o endereço físico inicial e o comprimento do segmento. O deslocamento é então adicionado ao endereço físico inicial para determinar o endereço físico final na memória. Se o deslocamento for maior que o comprimento do segmento, ocorre uma falha de segmentação.

Leonardo Vecchi Meirelles - 12011ECP002
Cap. 17 - Questão 1)

Uma estrutura de dados comum é a ~~lista~~ lista livre, que é uma lista encadeada de blocos livres de memória. Cada bloco livre contém um cabeçalho e um rodapé que armazenam informações sobre o tamanho do bloco e se ele está livre ou alocado.

Outra estrutura de dados é o vetor de bits, que é um bitmap que representa o status de alocação de blocos na memória. Cada bit corresponde a um bloco de memória, e um bit 0 representa um bloco livre enquanto um bit 1 representa um bloco alocado.

Uma estrutura de dados mais avançada é o buddy system, que é uma árvore binária de blocos livres que são potências de 2 em tamanho. Cada nó na árvore representa um bloco livre e seus filhos representam duas metades desse bloco. Essa estrutura é mais eficiente do que a lista livre ou vetor de bits para certos tipos de padrões de alocação.

Leonardo Vecchi Meirelles - 12011ECP002
Cap 17 - Questão 2)

Tipicamente existem 3 locais para armazenar a lista de livres:

- Como uma estrutura de dados separada na memória;
- Nos cabeçalhos dos blocos de memória alocados;
- No espaço não utilizado dentro dos blocos de memória alocados.

Independentemente de onde a lista livre esteja armazenada, o sistema de gerenciamento de memória deve ser capaz de manipulá-la com eficiência para alocar e liberar memória. Isso normalmente envolve operações como adicionar blocos à lista livre, remover blocos e mesclar blocos livres adjacentes.

Leonardo Vecchi Meirelles - J201J ECP002

Cap J7 - Questão 3)

Alguns dos algoritmos comumente usados são:

- First-Fit: procura o primeiro bloco de memória disponível que seja grande o suficiente para acomodar o tamanho solicitado.
- Best-Fit: procura o menor bloco de memória disponível que seja grande o suficiente para acomodar o tamanho solicitado.
- Worst-Fit: procura o maior bloco de memória disponível e aloca o processo para esse bloco.
- Buddy System: baseia-se na divisão da memória em pedaços menores chamados de "buddies". Tal processo foi melhor explicado na questão 1 deste capítulo.
- Slab allocation: projetado para uso com objetos pequenos e frequentemente alocados, como estruturas de dados ou buffers de rede.