Avaliação final

Juciane de Azevedo Chianca

Capítulo 13

1 – questão 4, página 10 - Uma vez detectado um impasse, quais as abordagens possíveis para resolvê-lo? Explique-as e comente sua viabilidade.

Quando ocorrer um impasse, o sistema deve detectá-lo, determinar quais as tarefas e recursos envolvidos e tomar medidas para desfazê-lo. A resolução de impasse pode ser feita de duas formas:

eliminar tarefas - uma ou mais tarefas envolvidas no impasse são eliminadas, liberando seus recursos para que as demais tarefas possam prosseguir.

retroceder tarefas - uma ou mais tarefas envolvidas no impasse têm sua execução parcialmente desfeita (uma técnica chamada rollback), de forma a fazer o sistema retornar a um estado seguro anterior ao impasse.

A resolução de impasses é relativamente pouco usada fora de situações muito específicas, porque as alternativas de resolução mencionadas sempre implicam perder tarefas ou parte das execuções já realizadas.

2 – questão 8, página 11 - Nos grafos de alocação de recursos da figura a seguir, indique o(s) ciclo(s) onde existe um impasse:

Na figura 1 o impasse é registrado nos ciclos t1 – r2 – t2 e t2 – r1 – t1

Na figura 2 o impasse é registrado nos ciclos t1 – r2 – t2, t2 – r3 – t3/t4 e t3 – r1 – t1

3- questão 9, página 11 - A figura a seguir representa uma situação de impasse em um cruzamento de trânsito. Todas as ruas têm largura para um carro e sentido único. Mostre que as quatro condições necessárias para a ocorrência de impasses estão presentes nessa situação. Em seguida, defina uma regra simples a ser seguida por cada carro para evitar essa situação; regras envolvendo algum tipo de informação centralizada não devem ser usadas.

Exclusão mútua - Os automóveis em sentidos diferentes não podem acessar o mesmo cruzamento ao mesmo tempo;

Posse e espera - Os automóveis que estão em um cruzamento solicitam acesso ao outro cruzamento sem liberar o primeiro.

Não-preempção - Os automóveis que ocupam um cruzamento só irão liberá-lo, quando poderem voltar a seguir na sua rota.

Espera circular - Os automóveis vermelhos dependem da liberação do cruzamento que os automóveis azuis estão ocupando, os automóveis azuis dependem da liberação do cruzamento que os automóveis verdes estão ocupando, os automóveis verdes dependem da liberação do cruzamento que os

automóveis amarelos estão ocupando, os automóveis amarelos dependem da liberação do cruzamento que os automóveis vermelhos estão ocupando.

Regra simples para evitar a situação: Os automóveis de qualquer sentido só devem avançar quando os dois cruzamentos do seu sentido estiverem liberados.

Capítulo 14

1 – Explique em que consiste a resolução de endereços nos seguintes momentos: codificação, compilação, ligação, carga e execução.

Codificação - o programador escolhe o endereço de cada uma das variáveis e do código do programa na memória.

Compilação – ao traduzir o código-fonte, o compilador escolhe as posições das variáveis na memória. Para isso, todos os códigos-fontes necessários ao programa devem ser conhecidos no momento da compilação, para evitar conflitos de endereços entre variáveis em diferentes arquivos ou bibliotecas.

Ligação – o ligador (linker) pega os arquivos objetos com suas tabelas de símbolos, define os endereços de memória dos símbolos e gera o programa executável.

Carga – um carregador (loader) é responsável por carregar o código do processo na memória e definir os endereços de memória que devem ser utilizados.

Execução – os endereços emitidos pelo processador durante a execução do processo são analisados e convertidos nos endereços efetivos a serem acessados na memória real.

2 - Como é organizado o espaço de memória de um processo?

Text, data, bss, heap, stack.

Text - contém o código binário a ser executado pelo processo, gerado durante a compilação e a ligação com as bibliotecas e armazenado no arquivo executável.

Data - contém as variáveis estáticas inicializadas, ou seja, cujo valor inicial é declarado no código-fonte do programa.

BSS - contém as variáveis estáticas não-inicializadas. Esta seção é separada da seção DATA porque as variáveis inicializadas precisam ter seu valor inicial armazenado no arquivo executável, o que não é necessário para as variáveis não-inicializadas.

Heap - esta seção é usada para armazenar variáveis alocadas dinamicamente, usando operadores como malloc(), new() e similares.

Stack - esta seção é usada para manter a pilha de execução do processo.