Lista 9 Sistemas Operacionais

Nome: William Cardoso Barbosa

1. Defina o que é uma aplicação concorrente e dê um exemplo de sua utilização.

   É uma aplicação estruturada de maneira que partes diferentes do código do programa possam executar concorrentemente. Este tipo de aplicação tem como base a execução cooperativa de múltiplos processos ou threads, que trabalham em uma mesma tarefa na busca de um resultado comum.

2. Considere uma aplicação que utilize uma matriz na memória principal para a comunicação entre vários processos concorrentes. Que tipo de problema pode ocorrer quando dois ou mais processos acessam uma mesma posição da matriz?

   Caso não haja uma gerência no uso concorrente dos recursos compartilhados, inconsistências nos dados podem ocorrer.

3. O que é exclusão mútua e como é implementada?

   É impedir que dois ou mais processos acessem um mesmo recurso simultaneamente. Para isso, enquanto um processo estiver acessando determinado recurso, todos os demais processos que queiram acessá-lo deverão esperar pelo término da utilização do recurso.

4. Como seria possível resolver os problemas decorrentes do compartilhamento da matriz, apresentado anteriormente, utilizando o conceito de exclusão mútua?

   Garantindo na aplicação que somente um único processo pode estar acessando a matriz por vez.

5. O que é starvation e como podemos solucionar esse problema?

   Starvation é a situação onde um processo nunca consegue executar sua região crítica e, conseqüentemente, acessar o recurso compartilhado. A solução para o problema depende de estabelecimentos de mecanismos de acesso pelo sistema operacional que garantam o acesso ao recurso por todos os processos que solicitarem uso.

6. O que é espera ocupada e qual o seu problema?

   Espera ocupada é um modelo de programação paralela caracterizado por testes repetidos de um condição que impedem o progresso de um processo e que só pode ser alterada por outro processo. Possui a grande desvantagem de levar a desperdícios de tempo em um monoprocessador, já que este passa parte do tempo testando condições (cujo resultado é falso), ao invés de realizar trabalho útil. Porém, pode ser uma solução aceitável para multiprocessadores.

7. Explique o que é sincronização condicional e dê um exemplo de sua utilização.

   Sincronização condicional é quando o acesso a um recurso compartilhado exige uma sincronização entre os processos; Exemplo clássico é a comunicação entre dois processos através de operações de gravação e leitura em um buffers.

8. Explique o que são semáforos e dê dois exemplos de sua utilização: um para a solução da exclusão mútua e outro para a sincronização condicional.

   Semáforos são mecanismos que resolvem o problema de exclusão mútua. Um semáforo pode ser visto como um objeto que pode sofrer dois tipos de operação sobre ele: trancando e destrancando a execução de instruções (p. ex., operações UP e DOWN, P e V). As operações sobre um semáforo são atômicas.

   Semáforos são implementados no sistema operacional e são considerados uma forma de IPC (semáforos também podem ser usados para sincronização tão bem como para obtenção de exclusão mútua). Da mesma maneira que SO's diferentes implementam versões diferentes de memória compartilhada e filas de mensagems, há várias implementações de semáforos. O POSIX.1b implementa semáforos com identificadores e sem identificadores. O System V também implementa semáforos e são estes os que vão ser usados neste experimento.

9. Apresente uma solução para o problema dos filósofos que permita que os cinco pensadores sentem à mesa, porém evite a ocorrência de starvation e deadlock.

   A classe *Philosopher* implementa as duas tarefas executadas por qualquer filósofo descrito no contexto de nosso problema. Podemos imaginar então, de forma bem simples, que um programa para fazer essas tarefas deve apenas tentar pegar um garfo qualquer. Caso não consiga, fique aguardando. Em seguida faz o mesmo

para o segundo garfo. Uma vez com os garfos nas mãos começa a comer. Ao final devolve um garfo e depois o outro.

O grande problema nesse algoritmo é que existem momentos em que se **todos** os filósofos pegarem **um garfo**, todos irão ficar **parados para sempre** aguardando o segundo garfo ficar disponível; gerando assim um ***Deadlock*** ou **Impasse!** Ou seja, não é uma **solução completa**.

10. Explique o que são monitores e dê dois exemplos de sua utilização: um para a solução da exclusão mútua e outro para a sincronização condicional.

    Monitores são mecanismos de sincronização de alto nível que torna mais simples o desenvolvimento de aplicações concorrentes.

    Na exclusão mútua, a implementação utilizando monitores não é realizada diretamente pelo programador; as regiões críticas são definidas como procedimentos no monitor e o compilador garante a exclusão mútua; a comunicação do procedimento e monitor é executada através de chamada a procedimentos e dos parâmetros passados.

    Na sincronização condicional, poder implementada utilizando-se de monitores através de variáveis especiais de condição; é possível associar a execução de u procedimento que faz parte do monitor a uma determinada condição.

11. Qual a vantagem da forma assíncrona de comunicação entre processos e como esta pode ser implementada?

    A vantagem deste mecanismo é aumentar a deficiência de aplicações concorrentes. Para implementar essa solução, além da necessidade de buffers para armazenar as mensagens, devem haver outros mecanismos de sincronização que permitam ao processo identificar se uma mensagem já foi enviada ou recebida.

12. O que é deadlock, quais as condições para obtê-lo e quais as soluções possíveis?

    Deadlock é a situação em que um processo aguarda por um recurso que nunca estará disponível ou um evento que não ocorrerá. Para que ocorra a situação de deadlock, quatro condições são necessárias simultaneamente:

    Exclusão mútua: cada recurso só pode estar alocado a um único processo em um determinado instante;
    Espera por recurso: um processo, além dos recursos já alocados, pode estar

esperando por outros recursos;
Não-preempção: um recurso não pode ser liberado de um processo só porque outros processos desejam o mesmo recurso;
Espera circular: um processo pode ter de esperar por um recurso alocado a outro processo e vice-versa.

Para prevenir a ocorrência de dedalocks, é preciso garantir uma das quatro condições apresentadas, necessárias para sua existência, nunca se satisfaça. A prevenção de deadlocks evitando-se a ocorrência de qualquer uma das quatro condições é bastante limitada e, por isso, na prática não é utilizada. Uma solução conhecida como Algoritmo do Banqueiro (implementada com a presença das quatro condições) também possui várias limitações. A maior delas é a necessidade de um número fixo de processos ativos e de recursos disponíveis no sistema. Essa limitação impede que a solução seja implementada na prática, pois é muito difícil prever o número de usuários no sistema e o número de recursos disponíveis.