**1) Por que é importante sincronizar processos em um sistema?** R: As mudanças feitas em um processo não são refletidas quando outro processo acessa os mesmos dados compartilhados. É necessário que os processos sejam sincronizados entre si, pois isso ajuda a evitar a inconsistência de dados compartilhados. Por exemplo: Um processo P1 tenta alterar dados em um determinado local de memória. Ao mesmo tempo, outro processo P2 tenta ler dados do mesmo local de memória. Assim, há uma alta probabilidade de que os dados sendo lidos pelo segundo processo estejam incorretos. **2) Diferencie Race Condition de Exclusão Mútua.** R: Durante a execução simultânea de processos, os processos precisam entrar na seção crítica em momentos para execução. Pode acontecer que, devido à execução de múltiplos processos ao mesmo tempo, os valores armazenados na seção crítica se tornem inconsistentes. Em outras palavras, os valores dependem da sequência de execução das instruções - também conhecida como race condition. A principal tarefa da sincronização de processos é se livrar das condições de corrida durante a execução da seção crítica. Isto é conseguido principalmente através da exclusão mútua. A exclusão mútua é uma propriedade da sincronização de processos que afirma que não podem existir dois processos na seção crítica em um determinado momento. Qualquer técnica de sincronização de processos utilizada deve satisfazer a propriedade da exclusão mútua, sem a qual não seria possível se livrar de uma condição racial. **3)** Explique qual o significado do termo espera ocupada? Que outros tipos de espera existem em um sistema operacional? A espera ocupada pode ser completamente evitada? Explique sua resposta. R: A espera ocupada é uma técnica de sincronização de processos na qual um processo/tarefa espera e verifica constantemente se uma condição deve ser satisfeita antes de proceder com sua execução. Na espera ocupada, um processo executa instruções que testam para que a condição de entrada seja verdadeira, como a disponibilidade de um cadeado ou recurso no sistema de computador. A espera ocupada não pode ser evitada por completo. Alguns eventos não podem desencadear um despertar; por exemplo, no Unix um processo não pode "dormir até que um arquivo seja modificado", porque o sistema operacional não fornece nenhum mecanismo para despertar automaticamente o processo quando o evento ocorre; alguma quantidade de pesquisas repetidas é necessária. **4)** Por que desabilitar interrupções pode não ser uma boa idéia para garantir exclusão mútua? Justifique sua resposta. R: Por que sempre que as interrupções forem desativadas, a CPU não poderá mudar de processo e os processos podem usar variáveis compartilhadas sem que outro processo acesse a CPU. Como eles podem usar variáveis compartilhadas, poderão utilizar a seção crítica mutuamente. **5)** O que são semáforos, e como eles funcionam? R: Semáforos são variáveis inteiras que são usadas para resolver o problema de seção crítica usando duas operações atômicas, espera e sinal que são usadas para sincronização de processos. A operação de espera decreta o valor de seu argumento S, se for positivo. Se S for negativo ou zero, então nenhuma operação é realizada. A operação do sinal incrementa o valor de seu argumento S. **6)** Quais as condições para a ocorrência de deadlock? R: Condição de exclusão mútua, Condição de posse e espera, Condição de não preempção e Condição de espera circular

**1)** Descreva o problema do jantar dos filósofos. R: Existem 5 filósofos sentados em uma mesma mesa de jantar circular, cada um tem seu próprio prato de espaguete e é necessário 2 garfos para 1 filósofo comer, porém, existe apenas 5 garfos na mesa entre cada par de prato. Os filósofos irão passar um tempo pensando e após isso irão comer, terminando de comer irão devolver os dois garfos a sua posição original e voltam a pensar. **2)** O código abaixo apresenta uma solução utilizando semáforos para solucionar o problema do Barbeiro. O problema do barbeiro consiste do seguinte: A barbearia possui um barbeiro, uma cadeira de barbeiro, e n (CHAIRS) cadeiras para os clientes que estão esperando para serem atendidos. Se não existem clientes para serem atendidos, o barbeiro senta na sua

cadeira e dorme. Quando um cliente chega, ele acorda o barbeiro para ser atendido. Se outros clientes chegarem enquanto o barbeiro estiver ocupado, ou eles sentam em alguma cadeira para clientes que esteja livre ou vão embora se a barbearia estiver cheia. Considerando esse problema, responda as seguintes questões: a. Descreva como funcionam os semáforos nesta solução. R: O mutex é um semáforo binário que irá proteger a região crítica; O customers são um semáforo inteiro de sincronismo, podendo assumir o número 0 para barbeiro dormindo, 1 para barbeiro acordado e clientes esperando. O barbers é semáforo inteiro de sincronismo podendo assumir 0 para barbeiro ocupado e 1 para barbeiro esperando b. Qual é a região crítica? R: A região crítica é quando o salão está cheio ou seja quando o waiting for maior que o número de cadeiras, no caso o waiting é a região crítica. c. O que aconteceria se os seguintes comandos fossem retirados: void barber -> down(&mutex); void customer -> up (&mutex); R: Provavelmente iríamos ter mais clientes do que cadeiras na barbearia devido ao semáforo de customers ter sido removido e o barbeiro poderá dormir para sempre pois o semáforo do barber foi retirado.

**1)** Discorra sobre a hierarquia de memória e as respectivas diferenças entre cada um dos tipos de memória. R: No projeto do sistema de computador, a Hierarquia da Memória é uma melhoria para organizar a memória de forma que possa minimizar o tempo de acesso. A Hierarquia de Memória foi desenvolvida com base em um comportamento de programa conhecido como localidade de referências. Este Projeto de Hierarquia de Memória está dividido em 2 tipos principais: 1) Memória Externa ou Memória Secundária - Composto de disco magnético, disco óptico, fita magnética, ou seja, dispositivos periféricos de armazenamento que são acessíveis pelo processador através do Módulo I/O. **2)** Memória Interna ou Memória Primária - Composto de memória principal, memória Cache e registros de CPU. Isto é diretamente acessível pelo processador. Os elementos presentes na hierarquia de memória são registradores( é a memória temporária usada pelo processador para realizar instruções), Memória ROM/EPROM(é nelas que se armazena instruções de inicialização do computador), Memória Ram(é a memória principal do computador, é nela que o processador irá endereçar os processos e instruções), Memória secundária ou HD( é usada para armazenar permanentemente), seguindo essa ordem a memória principal é a mais barata(baixo custo), com menor velocidade e com maior capacidade de armazenamento, enquanto os registradores são o completo oposto, sendo caros, rápidos e com pouca capacidade de armazenação 2) Descreva as responsabilidades do gerenciador de memória. R: Existem 5 responsabilidades que um gerenciador de memória pode assumir, Realocação, Proteção, Compartilhamento, Organização local e física. 1) Realocação: É utilizado para que o linker não escreva em um local indevido quando um programa na memória é linkado. 2) Proteção: É tomado o devido cuidado para que os processos não façam acessos indevidos.

3) Compartilhamento: Os processos que trabalham juntos em alguma tarefa podem precisar compartilhar o acesso à mesma estrutura de dados. O sistema de gerenciamento de memória deve, portanto, permitir a chamada regulada para área de memória compartilhada sem comprometer a proteção necessária. Portanto, esta exigência suporta proteção e controle de acesso. 4) Organização local: A memória principal em um sistema de computador é estruturada como um espaço de endereçamento linear, ou 1-Dimensional, compreendendo um arranjo de bytes ou palavras. A memória secundária, em seu nível físico, é igualmente estruturada. 5) Organização física: A memória do sistema é organizada em 2 níveis; um nível é a memória principal e o outro nível é a memória secundária. A memória principal oferece acesso mais rápido, mas tem um custo elevado e é uma memória volátil com menor capacidade de armazenamento.

**3)** Exemplifique a diferença entre a estratégia de partição fixa da estratégia de partição variável. Cite quais são as vantagens e desvantagens de cada estratégia.R:Em partições fixas o tamanho e número de partições são estáticos, sempre que temos que alocar uma memória de processo, então é encontrada uma partição livre que seja suficientemente grande para segurar o processo. Se não houver espaço livre disponível, o processo espera na fila para ser alocada a memória. Vantagens: Ele não utiliza a memória principal efetivamente, é mais fácil de implementar. Desvantagens: Apenas um processo pode ser colocado em uma partição, Há limitação no tamanho do processo, Há presença de fragmentação interna e externa, O grau de multi-programação é menor. Em partições variáveis o gerenciamento de memória é contínuo na qual a memória principal não é dividida em partições e o processo é alocado a um pedaço de memória livre que é grande o suficiente para que ela se ajuste. O espaço deixado é considerado como o espaço livre que pode ser utilizado por outros processos. Ele também fornece o conceito de compactação. Vantagens: O grau de multi-programação é maior, não há limitação no tamanho do processo, o processo é alocado em um pedaço de memória livre, utiliza eficazmente a memória principal. Desvantagens: É menos fácil de implementar

Pela definição de DEADLOCK: Um conjunto de processos do Sistema Operacional está em situação de Deadlock se todo processo pertencente ao conjunto estiver esperando por um evento que somente outro processo desse mesmo conjunto poderá fazer acontecer. Nas implementações irão ocorrer deadlocks assim que eles executarem o método down(&resource\_1) OU método down(&resource\_2) pois eles só irão chamar o método up referente ao mesmo resource no final de sua execução, para que eles não façam um deadlock seria necessário que após o método down ser concluído fosse chamado o método up referente ao down, liberando assim o resource para a outro função. Também há um problema no método use both, podendo ter um deadlock quando os dois tentarem acessá-lo ao mesmo tempo, seria necessário uma verificação de uso desse método