Perguntas:

1. Por que a multiprogramação é fundamental para a operação de um sistema operacional moderno?
   A multiprogramação permite que vários processos estejam prontos para serem executados ao mesmo tempo. Isso significa que, mesmo que um processo esteja aguardando alguma operação de entrada/saída, outros processos podem continuar sendo executados, garantindo que o sistema permaneça responsivo para o usuário.

2. Quais são os três estados principais em que um processo pode estar? Descreva sucintamente o significado de cada um.
   O processo pode está em execução, nesse estado ele está utilizando a CPU para executar instruções ativamente; O processo pode está em estado de espera, ou seja, ele está pronto para ser executado, porém não está na sua vez de utilizar a CPU; e por último ele pode está bloqueado, esse estado ocorre quando há algum tipo de comportamento que impossibilite a execução do processo.

3. Em todos os computadores atuais, pelos menos parte das rotinas de tratamento de interrupção é escrita em linguagem assembly. Por que?
   A linguagem assembly permite um controle preciso e direto sobre o hardware do sistema. As rotinas de tratamento de interrupção muitas vezes precisam lidar com questões de hardware em um nível muito baixo, como manipulação de registradores específicos do processador, acesso a dispositivos de E/S e alterações de estado crítico do sistema. Escrever essas rotinas em linguagem assembly proporciona um controle mais direto e eficiente sobre essas operações de baixo nível.

4. Qual é a diferença fundamental entre um processo e uma thread?
   A principal diferença entre um processo e uma thread está na independência e no isolamento dos recursos no caso de processos, enquanto as threads compartilham recursos e são executadas dentro do contexto de um processo.

5. Em um sistema com threads, existe normalmente uma pilha por thread ou uma pilha por processo? Explique.
   Em um sistema com threads, geralmente há uma pilha por thread, não uma por processo. Isso ocorre porque as threads compartilham o mesmo espaço de endereçamento de memória dentro de um processo

e, portanto, cada thread tem sua própria pilha separada, mas todas as pilhas residem dentro do mesmo espaço de endereçamento do processo.

6. O que é uma condição de corrida?
   As condições de corrida podem ocorrer em situações em que dois ou mais threads estão tentando acessar e manipular um recurso compartilhado ao mesmo tempo, sem a devida coordenação ou sincronização.

7. Dê um exemplo de condição de corrida que poderia ocorrer na compra de passagens aéreas por duas pessoas que querem viajar juntas.
   A condição de corrida poderia ocorrer quando houvesse apenas dois assentos no voo, dessa forma, eles terão que acessar ao mesmo tempo o sistema e escolher cada um, um dos assentos ainda disponíveis. Caso ocorra de uma outra pessoa escolher um dos assentos mais rápido que um dos dois, o processo falha.