Lista de Exercícios: Sistemas Operacionais

Assuntos: Gerenciamento de processos e Impasses.

1. Explique, com suas palavras, a diferença entre programa, processo e thread. Em seguida, descreva uma situação do cotidiano que poderia ser usada como analogia para explicar esses três conceitos a uma pessoa leiga.

2. Um processo passa pelos estados “Novo → Pronto → Execução → Bloqueado → Terminado”. Desenhe um diagrama de transições de estado e descreva em que situações reais (exemplo: leitura de arquivo, espera de entrada do usuário, preempção) cada transição ocorreria.

3. Explique o que é uma troca de contexto e por que ela é necessária. Depois, analise: por que trocas de contexto muito frequentes podem degradar o desempenho do sistema?

4. Em sistemas UNIX, cada processo pode gerar processos filhos com *fork()* e substituí-los com *exec()*. Explique o funcionamento dessas chamadas e discuta as vantagens e desvantagens do modelo hierárquico de processos em relação a um modelo totalmente plano (sem hierarquia).

5. Compare os principais mecanismos de comunicação entre processos (pipes, filas de mensagens, memória compartilhada e sockets). Para cada um, cite um exemplo de aplicação real em que ele seria a melhor escolha.

6. Um desenvolvedor afirma que “usar threads é sempre melhor do que usar processos, pois threads são mais leves”. Discuta criticamente essa afirmação, apresentando situações em que threads podem ser problemáticas e situações em que são ideais.

7. Explique o que é uma condição de corrida. Descreva um exemplo simples (pode ser em pseudocódigo ou em uma analogia) e explique como o uso de semáforos evita esse problema.

8. Escolha um dos problemas clássicos da IPC (Jantar dos Filósofos, Leitores e Escritores, Produtor-Consumidor).
a) Explique a essência do problema.
b) Proponha uma solução teórica, mencionando o mecanismo de sincronização que usaria.
c) Discuta possíveis limitações dessa solução em sistemas reais.

9. Dado o seguinte conjunto de processos com tempos de chegada e duração:

Processo	Chegada	Duração	Prioridade
P1	0	8	2
P2	1	4	1
P3	2	9	3
P4	3	5	2

a) Monte o gráfico de execução para os algoritmos FCFS e Round Robin (q=3).
b) Calcule o tempo médio de espera para cada um.
c) Analise qual algoritmo seria mais justo nesse caso e por quê.

10. Defina o que é um impasse (*deadlock*) e explique as quatro condições necessárias para que ele ocorra. Depois, cite um exemplo real (dentro ou fora da computação) que representa cada condição.

11. Considere os seguintes processos e recursos:

- P1 usa R1 e precisa de R2
- P2 usa R2 e precisa de R3
- P3 usa R3 e precisa de R1
  a) Modele o problema em um grafo de alocação de recursos.
  b) Verifique se há um impasse.
  c) Explique o raciocínio e proponha uma forma de resolver o problema.

12. Explique, em termos simples, a ideia por trás do algoritmo do banqueiro de Dijkstra. Depois, avalie: por que esse algoritmo é considerado impraticável em muitos sistemas modernos? Sugira uma situação em que ele seria útil..

13. Compare as estratégias de prevenção, evitação e detecção de impasses. Para cada uma, cite:

- um ponto forte,
- uma limitação prática,
- e um exemplo de uso em sistemas reais.

14. Em um sistema Linux moderno, múltiplos processos compartilham CPU, memória e dispositivos de E/S. Discuta como os mecanismos de gerenciamento de processos, escalonamento e sincronização atuam juntos para evitar impasses e garantir desempenho. Conclua avaliando como as decisões de projeto do kernel influenciam diretamente a experiência do usuário.